DETECTION AND TRACKING OF DIVERS FOR UNDERWATER HUMAN-ROBOT INTERACTION SCENARIOS

A Thesis
Presented to
The Academic Faculty

by

Kevin DeMarco

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2016

DETECTION AND TRACKING OF DIVERS FOR UNDERWATER HUMAN-ROBOT INTERACTION SCENARIOS

Approved by:

Dr. Ayanna M. Howard,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ayanna M. Howard, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Thomas R. Collins
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tucker Balch
College of Computing
Georgia Institute of Technology

Dr. Michael E. West
Georgia Tech Research Institute
Georgia Institute of Technology

Date Approved: 29 August 2016

For my family.

ACKNOWLEDGEMENTS

I would not have completed this dissertation without the advisement and encouragement from a large number of individuals. However, I would like to first thank my advisor, Dr. Ayanna Howard, who has helped me transform my way of thinking from simply solving the immediate engineering problem at hand to thinking about a problem in its research context. Without her guidance, I would have never made it this far. Also, I am grateful to Dr. Michael E. West for introducing me to the interesting world of Underwater Robotics, which, ultimately, led me to leaving my "day job" in pursuit of research problems. Dr. Tom Collins has provided a great deal of direction and support in terms of my academic research and on funded projects at the Georgia Tech Research Institute (GTRI). I was honored to have Dr. Magnus Egerstedt and Dr. Tucker Balch serve on my committee. I had the pleasure of taking courses at Georgia Tech from both of them and I think they are two of the finest and most engaging professors.

Of course, I have to thank my fellow HumAnS (Human-Automation Systems) Lab members. It was your suggestions and encouragements during group meetings and presentations that helped to form my dissertation topic.

I would also like to thank the researchers at GTRI's Aerospace, Transportation, and Advances Systems (ATAS) Laboratory. Through my graduate research assistantship at ATAS, I was able to hone my knowledge of behavior-based robotics and contribute to major programs under the guidance of several excellent technical managers: Lora Weiss, Michael Heiges, John Doss, Charles Pippin, Zsolt Kira, and Don Davis.

I am grateful to my family and friends for putting up with my non-deterministic work schedule and helping me take my mind off work.

Finally, I am thankful for Mira's support and encouragement throughout this entire process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The underwater domain is a dangerous and complex environment for human divers. Often, divers have to monitor their own life support systems as they navigate to the work site or operate dangerous machinery. Military divers have to navigate for extended periods of time without surfacing or without using localization techniques that might give away their positions. Human divers have operated under these harsh conditions for decades with few advancements in technology. In fact, a diver performs the basic task of navigation by aligning the body with a compass and counting leg kicks (i.e., human-oriented dead-reckoning). It is proposed that an Underwater Robotic Assistant (UWRA) will improve the efficiency and safety of the diver's underwater operations by providing several key capabilities. For example, the UWRA can provide navigation assistance, ferry tools from the surface, enter structures too dangerous for human divers, and carry hazardous materials. However, in unstructured environments, underwater robots are limited in their ability to localize and track a human diver at the resolution required to enable diver-robot interactions. Optical cameras can be rendered useless by the turbidity of the water, localizing radio signals do not propagate well through the water medium, and acoustic positioning systems can be expensive to deploy. We propose that by developing novel 2D imaging sonar processing techniques, an underwater robot can detect, track, and trail a human diver.

The objective of this research is to detect and track human divers in 2D imaging sonar data. While the physical properties of sonar allow it to detect objects at longer ranges than optical cameras in underwater scenarios, it is plagued with noise and multi-path propagation. Also, when a diver is ensonified with a 2D imaging sonar, a fragmented acoustic reflection is returned. The fact that a single object can produce multiple returns means that tracking the human diver cannot be solved by applying traditional multiple hypothesis tracking algorithms, which operate on the assumption of each object generating only a single measurement. To overcome the sonar noise and multiple fragmented returns, we

developed a novel adaptive thresholding algorithm and a hierarchical multiple object tracking algorithm. While the Kalman filter is extensively used in our tracking algorithms, we developed a novel method for adaptively modifying the Kalman filter's measurement matrix to track objects that generate multiple measurements.

Since a 2D imaging data set of divers did not exist when we began our work, we had to generate our own data set. We accomplished this by both integrating a 2D imaging sonar on our own underwater robot and collaborating with other researchers that developed their own underwater robot. We evaluated the effectiveness of our image processing and tracking algorithms on the data set we collected. For evaluation purposes, ground truth was provided by hand-annotating the sonar data sets that we collected.

CHAPTER I

INTRODUCTION

The underwater domain is a dangerous and complex environment for human divers. Often, industrial divers have to monitor their own life support systems as they navigate to the work site or operate dangerous machinery. The dangers that industrial divers face is highlighted by the fact that the rate of death for industrial divers is 40 times higher than the national average death rate for all workers [16]. In the military domain, military divers have to navigate for extended periods of time without surfacing or without using localization techniques that might make their positions known. Human divers have been operating under these harsh conditions for decades with few advancements in technology. In fact, the basic task of navigation is accomplished by the diver using a compass and counting leg kicks (i.e, human-oriented dead-reckoning). There have been some attempts to design underwater GPS-like systems, but they require the diver to tow a line attached to the surface or the setup of an expensive acoustic network [14] [43]. It is proposed that an Underwater Robotic Assistant (UWRA) will improve the efficiency of the diver's underwater operations by providing several key capabilities. For example, the UWRA can provide navigation assistance, ferry tools from the surface, enter structures too dangerous for human divers, and carry hazardous materials. Eventually, the UWRA could become a diving "buddy," taking on the role usually filled by a human. The UWRA would be responsible for monitoring the human diver's health, assisting in navigation, and providing additional air if the human diver's breathing system failed. However, there are three major impediments to the implementation of such a UWRA: tracking the diver, communicating with the diver, and interacting with the diver. In this research, we focus on the first impediment. The diver detection and tracking problem is difficult underwater because optical cameras cannot be used reliably in many situations due to water turbidity. In fact, we encountered this issue in one of our early field experiments when we had planned on using optical methods for diver

detection. Three still images taken from a remotely operated vehicle's (ROV) camera are shown in Fig. 1. The image data was captured in Lake Lanier, which is a lake in the U.S.

(a) 2 m from swimmer. (b) 1.5 m from swimmer. (c) 0.5 m from swimmer.

Figure 1: Optical visibility of swimmer in Lake Lanier.

state of Georgia. The swimmer is clearly visible at distances of less than 0.5 m. However, at distances over 2 m, the diver is almost entirely unrecognizable. The UWRA would be forced to remain with this limited visibility range of the diver or risk losing the diver in the murky water. This would limit the UWRA's ability to perform tasks that require the UWRA to fetch items or search for the diver. Even though this is only a single example of the shortcomings of underwater optical detection, it would be naive to assume that an optical diver detection method would be viable in freshwater, saltwater, lake, ocean, and man-made tank underwater environments. While recreational scuba divers have the luxury of diving in environments with high-visibility, industrial and military divers have to dive where their work is required. For this reason, we decided to explore the use of 2D imaging sonars for the detection and tracking of divers. Unlike an optical sensor, 2D imaging sonar is not attenuated by highly turbid water. Also, commercial 2D imaging sonars that operate in the 900 kHz band can have a maximum range of 100 m [13]. Of course, 2D imaging sonars do have some flaws. First of all, the same physical properties of acoustics that allow the sonar to detect objects at longer-ranges also contribute to multi-path interference at the sonar head. This can result in "ghost images" appearing in the sonar data. Also, compared to optical images, sonar images are highly susceptible to salt-and-pepper noise. Finally, different materials produce different levels of acoustic return when ensonified with acoustic energy. Thus, when ensonified, a diver's acoustic signature is characterized by multiple fragmented returns. An example of the scuba diver's fragmented returns is provided in Fig.

(a) Scuba Diver in 2D imaging sonar frame. (b) Scuba Diver's fragmented returns.

Figure 2: A scuba diver's acoustic signature is characterized by multiple fragmented returns.

2a, where returns representing the diver's hands, feet, and abdomen are visible. However, the diver's body is not completely connected, spatially. The diver's fragmented returns are made more apparent after image processing techniques are used to separate the image into blobs, as shown in Fig. 2. Despite these shortcomings of 2D imaging sonar, the ability to detect objects and divers at longer distances than optical sensors makes 2D imaging sonar an attractive sensor for enabling Underwater Human-Robot Interaction. It is the objective of this research to apply image processing and tracking techniques to 2D imaging sonar data to overcome these shortcomings and make 2D imaging sonars a viable option for collaborative diver detection and tracking. This leads us to our thesis statement:

Multiple fragmented object tracking algorithms can be used to detect, track, and classify swimming scuba divers from an underwater robot equipped with a 2D imaging sonar, providing a robust and affordable approach to detecting scuba divers compared to using current diver detection methods.

It is worth noting that a straightforward method of detecting a diver could be to place an acoustic beacon on the diver that the UWRA could track. This method would be a worthwhile engineering effort, but is limited in potential. By using 2D imaging sonar to detect and track divers, we also implicitly detect and track other non-diver objects in the environment. This improves the UWRA's situational awareness and opens up avenues for future research in 2D imaging sonar processing.

Even though it is very clear that the objective of our research is to detect and track divers in 2D imaging sonar, we did not immediately arrive at this objective. Our overall

objective was to explore the new domain of Underwater Human-Robot Interaction (UHRI). We wanted to study how human divers communicate and work together to accomplish technical tasks and then apply those methodologies to a human-robot diver team. We even conducted a UHRI case study in which we devised a system for using an ROV to communicate information between a team of engineers at the surface and a submerged scuba diver. However, we quickly learned that just being able to locate and track the scuba diver was a difficult task for a robot. Most of the research in collaborative diver detection focused on optical detection, while the detection of adversarial divers relied on passive acoustic sensors and long-range active sonars that only returned a single "blip" for possible diver targets. We wanted to explore the use of novel image processing and tracking algorithms to detect and track collaborative divers with 2D imaging sonars, which produce multiple fragmented "blips" for each possible diver. By developing these diver detection algorithms we would be enabling future research in UHRI. Thus, we included a chapter on the UHRI case study that we conducted to provide context for the diver detection and tracking algorithms that we developed.

Through the development of our diver detection and tracking algorithms, we produced four primary contributions:

1. Developed an adaptive thresholding algorithm for 2D imaging sonar data that outperforms static and gradient-based thresholding algorithms in terms of Receiver Operating Characteristic (ROC) analysis.

2. Architected a hierarchical multiple object tracker that combines the benefits of Munkres assignment algorithm for low-level blob tracking and Kalman filters for high-level object tracking.

3. Designed and implemented a method for adaptively modifying the Kalman filters measurement matrix, R, to account for an object generating multiple fragmented returns when ensonified.

4. Constructed a diver model for classifying a track object as a diver or not by attempting to track the divers fins. If the confidences of the fin trackers are high, then the object

is classified as a diver. Our fin tracker classifier outperformed a baseline velocity-based classifier.

Our contributions make use of novel image processing and multiple hypothesis tracking techniques. Since a publicly available 2D imaging sonar data set of divers does not exist, we had to generate our own data set. This was accomplished by both integrating a 2D imaging sonar on our own underwater robot and working in collaboration with researchers at the Naval Postgraduate School (NPS) that developed their own underwater robot for UHRI. We collected the sonar data set through several field trials in varying conditions. The effectiveness of our image processing and tracking algorithms was evaluated on the data sets that we collected, where ground truth was provided through hand-annotation.

The remainder of this dissertation is organized as follows:

Chapter 2: The related work and motivation for our problem of detecting and tracking collaborative divers.

Chapter 3: A description of the underwater robotic platforms that we used to collect our sonar data set. Also, a description of the data collection locations and conditions is provided.

Chapter 4: A case study of using an underwater robot to enable communication between a team of engineers at the surface and a submerged diver.

Chapter 5: A description of the image processing techniques that were used to reduce noise in the sonar data and prepare the data for the tracking pipeline. We also describe our novel adaptive thresholding algorithm.

Chapter 6: The hierarchical multiple object tracking algorithm is presented. We discuss how we form image pixels into low-level blob tracks, which are then formed into higher-level object tracks.

Chapter 7: Our novel concept of classifying generic sonar objects as divers is presented. We show that sonar objects can be classified by evaluating the confidence levels of fin trackers in the vicinity of longer-lived tracks.

Chapter 8: Concluding remarks and comments about how this work can enable progress in the new field of UHRI.

CHAPTER II

RELATED WORK

The field of Underwater Human-Robot Interaction (UHRI) is still in its infancy, but there are a number of researchers that have started to operate in this domain. This domain encompasses the sensors, algorithms, and autonomy required for a UWRA to facilitate a human's underwater tasks. Researchers in this domain have focused primarily on developing algorithms that enable a UWRA to detect, communicate, and physically interact with a human diver.

2.0.1 Diver Detection

The diver detection literature is generally segmented into two main fields: detection of a potential underwater adversary using a network of sensors for security purposes and detection of a cooperative diver using sensors on or near a robotic assistant. An underwater adversary could be a free swimmer, a diver equipped with scuba equipment, a diver equipped with a rebreathing system, or a diver using a Diver Propulsion Vehicle (DPV). Since divers using scuba gear exhale gas bubbles that are visible at the surface, divers that wish to maintain stealth use rebreathing systems to store exhaled gases within the tanks they carry. Adversarial detection in urban harbor areas is often restricted to passive-sonar systems due to reverberation caused by shallow waters and restrictions on the use of active-sonar near marine life [38]. However, in [25], researchers used a network of passive acoustic "tripwires," active acoustic sensors, and magnetic sensors to detect and track a diver in a harbor. For the detection of closed-circuit scuba divers that use rebreathing systems, researchers tuned the passive-sonar systems' filters to match the acoustic signature of the diver's breathing apparatus. In a closed-circuit system, the primary source of acoustic emission is the decompressing and mixing of gases during inhalation[38]. DPV systems also generate a significant amount of prop noise that can be detected with passive acoustic systems.

While much of the diver detection literature has focused on detecting adversarial divers in harbor settings, in this research, diver detection refers to detecting a cooperative diver from the perspective of a UWRA. For detection at short-ranges, researchers have used optical cameras to track human divers. Researchers at McGill University have used frequency analysis in video data to detect the periodic motion of a human diver's fins [51]. After the presence of a human-specific frequency was detected in the video data, an Unscented Kalman Filter (UKF) was attached to the diver for robust trajectory tracking, such that an underwater robot could follow the human diver. While their optical tracking algorithm worked well in close-proximity to the underwater robot and in favorable underwater lighting conditions, a robust UWRA performing track-and-trail will require additional sensors. More recently, researchers have experimented with optical video stream registration to facilitate diver detection from a moving robotic platform [15]. Specifically, the Fourier Mellin Invariant (iFMI) registration method was implemented and tested on a series of videos that were posted to the Internet under the Creative Commons license. Unfortunately, both of these diver detection methods relied upon monocular optical vision, which can may not perform well in low-visibility or high-turbidity environments.

While acoustic networks are often used to detect nefarious divers, they can also be used to detect and track cooperating divers, while providing the diver navigation information [58]. A major drawback to an underwater acoustic network for diver localization is the high-cost of deploying such a network. Long baseline (LBL) acoustic positioning has been used to track Autonomous Underwater Vehicles (AUV) in under-ice and open water settings. However, LBL positioning could also be used to track a diver equipped with an acoustic beacon. An LBL network determines the positions of objects equipped with acoustic beacons through a time-of-flight calculation and triangulation across multiple moored acoustic beacons. While LBL networks have proven to be effective, again they suffer from the high-cost associated with configuring and deploying a static acoustic network. Meanwhile, the ultrashort baseline (USBL) system can be easily deployed because it only requires a transmitter on the object to be tracked and a transducer on the object that is performing the tracking. However, USBL systems suffer from low update rates, low-precision at long-ranges, and multipath

issues which result in inaccurate position information [36].

The first stages of the detection of a human diver from high-frequency forward-looking sonar images rely heavily upon computer vision techniques that are closely associated with the filtering of gray scale video data and motion detection. The literature on these topics is extensive and is often related to the background subtraction methods used to detect motion in surveillance video. Piccardi provided an excellent survey of background subtraction techniques, of which the Running Gaussian and Gaussian Mixture Model (GMM) methods were utilized in this research [45, 56].

2.0.2 Data Association & Tracking

The literature associated with the data association and tracking fields is vast. At the heart of many tracking applications is the Kalman filter, which provides a method for computing optimal state estimates and confidences associated with those estimates as new measurements are consumed [33]. However, deciding which measurements should be associated with various Kalman-like filters is related to the data association problem. One solution to the data association problem is to maintain a list of possible tracks by enumerating the ways in which each measurement can be associated with each possible target. This method is typically implemented with multiple hypothesis trees and is called multiple hypothesis tracking (MHT) [47]. Unfortunately, without using heuristic methods to prune the multiple hypothesis tree, computational complexity quickly becomes an issue. Still, MHT has a solid track record of being implemented in many applications [2, 57, 12].

The Probabilistic Data Association Filter (PDAF) and the Joint Probabilistic Data Association Filters (JPDAF) are other methods that have gained in popularity since MHT was first developed [6]. The PDAF and JPDAF both make use of Kalman filters for track-gate validation and state estimation, but they differ from MHT in how they associate measurements with tracks. Instead of keeping track of every possible measurement-to-track association, the PDAF combines multiple weighted measurements that fall within the track's validation region. This is known as making a "soft" decision. Opposed to a "hard" decision, which would assign one measurement to one track. The JPDAF is the multiple

track extension of the PDAF. The PDAF and JPDAF are known to have superior execution times to traditional MHT implementations due to the lack of tracking multiple hypotheses [5]. The PDAF has been used to track adversarial divers [50]. This was accomplished by detecting the diver's exhaled gas bubbles with 90 kHz sonar and tracking the bubble detections with a PDAF. A bubble model was developed behind the target, such that the trailing bubbles did not dominate the target's state estimates.

While not directly related to detecting divers in sonar, there has been a great deal of research in human detection in laser range data for the purpose of having a robot follow a walking person [17, 35]. Laser range finders and 2D imaging sonars are similar in that both sensors return multiple measurements for each object. This is opposed to traditional radar and the 90 kHz sonar previously described, which typically returned single points for each measurement. People can be modeled as a high-level track consisting of two legs, which are tracked independently in laser range data [4].

2.0.3 Underwater Human-Robot Interaction

Since radio waves are highly attenuated in water, early work with underwater robots involved teleoperation via a tether to the surface. A remotely operated vehicle (ROV) operator directly controlled an ROV's thrusters and manipulators while monitoring the ROV's camera, sonar, and other sensor measurements on a screen. Teleoperating an underwater robotic system is relevant to the field of Human-Robot Interaction (HRI) since HRI is concerned with human control of robotic systems and synchronizing multiple data sources and frames of reference to provide a cohesive portrayal of the environment to the human operator [28]. Complex underwater operations, such as underwater construction and archaeological survey, are still conducted by a human operator teleoperating an ROV. To reduce the possibility of human error, autopilots and higher-level waypoint path-following functions have been implemented on ROV systems [60]. Also, force-feedback joystick control and joystick input-filtering have been used to assist the operator in understanding the forces being applied to the ROV and operating actuators within their safe-ranges [54]. Another main thrust in UHRI has been the design of virtual reality environments to provide a cohesive image of

the environment for the operator [60]. Virtual reality systems help the operator quickly identify manipulator configurations, obstacles, and underwater landmarks. By overlaying digital imagery and 3D models on live video data, researchers have also developed underwater augmented reality systems [20]. Since tethered ROVs are widely used in the maritime community, there has been a great deal of work in improving the underwater teleoperation interface. Still, there is plenty of unexplored research related to the human-robot interaction between an underwater robot and a co-located underwater diver.

In the Underwater Human-Robot Communication (UHRC) domain, the authors of [52], developed the *RoboChat* language, which allowed a diver to provide commands to a UWRA through fiducial tags. The researchers performed tests to assess the cognitive load required for a diver to use *RoboChat*, while still monitoring critical health systems [22]. While industrial and military divers typically use hand gestures and body position to convey meaning, the *RobotChat* language makes use of a series of fiducial tags that program a series of commands into the robot. *RoboChat* is an innovative solution to the underwater communication problem, but it is not necessarily a natural means of communication for a typical diver. Indeed, a primary tenant of HRI is to develop interactions between humans and robots that do not require the human to greatly diverge from previously accepted modes of operation.

Though not completely suited to the unstructured environments associated with field robotics, the Swimoid was created to enable UHRI in a swimming pool setting [61]. The Swimoid is an underwater vehicle with wheels and thrusters that can continuously position itself under a swimmer and provide feedback to the swimmer concerning swimming form through its LCD interface. This enables a swimming coach to help improve a swimmer's stroke while the swimmer is actually swimming, which is often difficult due to water in the swimmer's ears.

This problem of UHRI has recently gained attention from academic universities in Europe because of the Cognitive Autonomous Driving Buddy, or CADDY, program [37]. CADDY is a newly sponsored EU funded project that focuses on developing symbiotic links between the diver and maritime robots. Various universities throughout Europe are

involved with the program and will specialize in underwater 3D mapping, diver detection, vehicle development, and various other topics associated with maritime robotics. The CADDY system is composed of an underwater remotely operated vehicle that is tethered to an autonomous surface vessel. Diver detection in the CADDY system consists of attaching an acoustic beacon to the diver within a USBL system.

2.0.4 Navigation and Control of UWRAs

As with most robotic fields, the field of UHRI can be segmented into three major areas: control systems, sensing, and autonomy. This work is most closely associated with underwater autonomy. Specifically, this research focuses on the autonomous functions that are required to enable human-robot interaction; thus, allowing a human and robot to cooperatively complete a task. Still, there are a number of topics related to control and sensing that must be solved before a full system is deployed. One of the goals of a UWRA is to perform tool hand-off with a human diver. However, if not properly accounted for, the added mass of the tool can cause the robotic system to become negatively buoyant and unstable. Thus, researchers are experimenting with adaptive control strategies for underwater vehicles, such that the vehicles can interact with the environment by carrying tools or scientific samples [63].

Since monocular vision has proven to be such a difficult problem, even in structured above-water scenarios, it seems doubtful that a monocular vision solution will suffice underwater. For ship-hull inspection, researchers have equipped a remotely operated vehicle (ROV) with a stereo-vision camera [41]. The stereo-vision camera was used not only for inspecting the hull of the ship, but also for ROV localization. Stereo-vision systems have also been implemented on diver-portable platforms that allow the diver to create a 3D map of the ocean floor or an underwater archaeological site [31] [7]. It is conceivable that a similar stereo-vision system could be used to detect a diver at close-ranges.

2.0.5 Underwater Human-Robot Interaction Scenarios

Currently, underwater diver operations involve such tasks as construction of underwater structures, destruction of underwater structures, ship salvage, search-and-rescue of personnel, placement of underwater cables, hazardous waste removal, and many other activities. In many industrial diving applications, the diver is tethered to the surface, so that the top-side support team can provide the diver with the appropriate mixture of gases to safely prolong the diver's bottom-time. However, in more unstructured missions, the diver's freedom from a tether is required, so that the diver can search the ocean floor for archaeological artifacts or enter underwater wrecks. In both diving situations, the diver could benefit from a robotic assistant for a number of tasks. For example, the diver cannot safely ascend quickly to the surface for fear of being a victim of the "bends." The "bends," or decompression sickness, occurs when a diver ascends to the surface at a rate that exceeds the rate at which the body can naturally remove excess nitrogen [49]. As Nitrogen comes out of solution in the bloodstream, the excess Nitrogen forms deadly bubbles at various locations in the human body. However, the "bends" would not affect a robotic assistant. When the diver breaks a tool or wants to immediately return an object to the surface, the UWRA could provide a safe means of transport to and from the surface.

Underwater navigation for human divers is a difficult skill to master. A diver relies on dead-reckoning for the majority of navigation by counting leg kicks and aligning the body with a compass [49]. However, this method of navigation does not take into account the effect of ocean currents on the diver's movement. To counteract this effect, experienced divers will try to align the direction of travel with waves in the sand and remember important landmarks, such as rocks, reefs, and man-made artifacts. Obviously, diver-based navigation could be augmented with a UWRA that has the appropriate sensors for accurate underwater localization. Many ROVs and AUVs are equipped with Acoustic Long-Baseline (LBL) localization and there have been many advances in underwater Simultaneous Localization and Mapping (SLAM) [48]. If a method of communicating the current position from the robot to the diver is developed, the robot could surely assist the diver with various navigation-based tasks.

2.0.6 Underwater Communication

Tethered underwater robots have been used since the 1960's. Since radio communications are highly attenuated underwater, the only practical solution to controlling an underwater robot has been to use a waterproof cable to communicate actuator commands to the robot and sensor data to the operator. However, when scientists required technology that could map larger swaths of the ocean floor and provide persistent monitoring of scientific targets, untethered robotic systems were the solution. To accommodate this change, underwater robots required more autonomy and a means to wirelessly communicate through water. One of the primary methods of underwater robot communication was inspired by the methods used by scientific buoy systems that were used to measure water salinity and wave motion: acoustic communication. Acoustic communication involves packetizing digital information and modulating a transducer to induce an acoustic wave in the water medium. The Woods Hole Oceanographic Institute's (WHOI) MicroModem has successfully been used on the Bluefin and GTRI Yellowfin for high-level mission start, mission stop, and heart-beat information [19]. The WHOI MicroModem can perform in multiple digital communication modes, including frequency-hopping frequency-shift keying (FH-FSK) and phase-coherent shift keying (PSK) [27]. It can also function in a long baseline (LBL) acoustic navigation system. One of the strengths of acoustic communication, the long communication distances, is also one of its weaknesses. In shallow waters, the long-wavelengths associated with acoustic communications results in significant multi-path signal reception. Also, the long acoustic wavelengths also limit the data rate at which information can be exchanged.

Recently, underwater optical digital communication has become a reality and involves the high-speed modulation of visible light to transfer data [18]. Contrary to acoustic communication, where the long-wavelength signal can be detected for miles, optical communications require shorter distances and typically, line-of-sight. However, optical communications could provide the high-bandwidth, low-latency, limited multi-path communication method required for co-located UHRI. Having to maintain line-of-sight between a robot and diver is not an unrealistic requirement in a UHRI scenario. In fact, since the robot will be the diver's dive buddy in many situations, it will have to be within the required distance to

provide a secondary air supply if the diver's air supply fails.

2.0.7 Underwater Autonomy

As underwater robots have progressed from teleoperated machines to fully autonomous agents that can adapt to unforeseen obstacles, the onboard software autonomy has had to become more flexible. In a tethered system, the ROV operator can modify objectives, react to obstacles, and change the mission length based on the sensor data that is provided to the operator during mission execution. However, when the system becomes untethered, the human operator will not be able to have such fine grain control of the system due to the limited bandwidth issues associated with underwater communications. The Mission Oriented Operating System (MOOS) coupled with the Interval Programming Helm (IvP-Helm) was designed to provide autonomous functionality to maritime robots through behavior fusion [42]. While MOOS is a general robotics publish-and-subscribe communication middleware, similar to the Robot Operating System (ROS), IvP-Helm, was developed to provide a structure by which individual robotic behaviors could be developed and fused through Interval Programming [9]. For example, IvP-Helm could be configured to enable the track-and-trail of another vessel, while avoiding other vessels that come into close-contact with ownship. IvP-Helm does not work by scripting specific actions. Instead, the course and speed of ownship emerge as a result of the blending of various robotic behaviors. MOOS-IvP has successfully been deployed on autonomous kayaks, Bluefin, the GTRI Yellowfin, the King-Fisher, and other autonomous maritime vessels. While MOOS-IvP has successfully been deployed on maritime systems, MOOS-IvP has not been used in a scenario involving HRI. Typically, a mission within MOOS-IvP is initiated by a human on the shore, but then the autonomous vessel operates completely on its own until the mission is completed. In a UHRI scenario, the robot will have to communicate with a diver and react to the diver's intentions, physiology, and commands. The IvP-Helm behavior fusion engine is abstract enough to handle such scenarios, but additional modules and behaviors would have to be developed to provide the necessary HRI data inputs to IvP-Helm.

2.1 Diver Detection Methods Summary

In summary, there are a number of currently available methods of detecting and tracking a cooperating diver, which could be used to enable Underwater Human-Robot Interaction scenarios. However, they each have their own potential weaknesses. The diver detection methods and their associated drawbacks are tabulated in Table 1. Even though RADAR

Table 1: Diver Detection Methods

Diver Detection Method	**Weaknesses**
Low-frequency sonar (90 kHz)	Low-resolution tracking. Low-precision at long-ranges.
Long baseline (LBL)	High-cost of deploying acoustic network.
Ultrashort baseline (USBL)	Low-precision at long-ranges and high-multipath noise.
Optical camera	Susceptible to low-light / high-turbidity environments.
RADAR	Radio frequencies are highly attenuated in water.
LIDAR	Light is highly attenuated in water.

and LIDAR have not been specifically discussed, they are included for completeness. Since radio frequencies and light are highly attenuated by water, water and light penetrating systems require large power sources. The large power source requirement makes equipping a relatively small underwater robot with a RADAR or LIDAR system difficult. Our approach of detecting cooperating divers with 900 kHz 2D imaging sonar provides more robust detection across various environmental conditions than optical cameras and higher-resolution tracking than traditional low-frequency sonar systems.

CHAPTER III

DATA COLLECTION & UNDERWATER ROBOTIC PLATFORMS

3.1 Introduction

In order to test our diver detection and tracking algorithms, we had to acquire a data set. Unfortunately, since 2D imaging sonars have only recently become widely available, a freely available 2D imaging sonar data set did not exist, let alone a 2D imaging sonar data set containing divers. Thus, we had to generate our own 2D imaging sonar data of divers in various situations. Collecting the data set involved obtaining remotely operated vehicles (ROVs), modifying them to carry the 2D imaging sonar sensor, and operating the data collection robot in the field.

Sonar data was collected at the Georgia Tech Olympic dive pool, Lake Lanier, the Georgia Tech acoustic tank, and during the NASA NEEMO 20 mission. The recording hardware, methods, and diver equipment improved as the data collection sessions progressed. The primary contribution in this section is the collection of a series of data sets in which a diver was ensonified by a 2D imaging sonar and recorded with optical sensors, where applicable.

3.2 Underwater Robotic Platforms

During our data collection sessions, we used two different underwater robotic platforms: the VideoRay Pro 4 and the Naval Postgraduate School's ROV for Underwater Human-Robot Interaction.

3.2.1 VideoRay Pro 4

The VideoRay Pro 4, as shown in Fig. 3a, is a commercially available remotely operated vehicle (ROV) that is used for the inspection of underwater man-made structures and natural formations. The VideoRay is equipped with an NTSC (National Television System Committee) analog camera with approximately 570 lines of resolution. The camera is mounted on a tilt-servo that allows the camera to tilt vertically 160° [62]. Since underwater

(a) VideoRay Pro 4 with attached BlueView P900-45 and manipulator arm.

(b) The VideoRay Pro 4 with a scuba diver in the Georgia Tech Acoustic Dive Tank.

Figure 3: The VideoRay Pro 4 ROV and a sonar image collected with the ROV.

settings are often very dark due to the attenuation of light, the VideoRay is equipped with two LED lights that provide 3,600 lumens of lighting [62]. The VideoRay uses a configuration of three thrusters: two horizontal thrusters for movement within the X-Y plane and one vertical thruster for movement in the Z-direction. Since underwater wireless communications are inadequate for real-time teleoperation, a tether from the surface to the ROV allows a human operator to send commands over an RS-485 communication bus. A manipulator was also attached to the VideoRay to allow for underwater intervention.

3.2.2 Agile Close-Quarters Underwater Autonomous System (ACQUAS)

(a) The ACQUAS is equipped with a sonar, optical camera, Doppler velocity log, manipulator, and inertial navigation system.

(b) The ACQUAS transporting a scientific sample to an Aquanaut during the NASA NEEMO Mission 20.

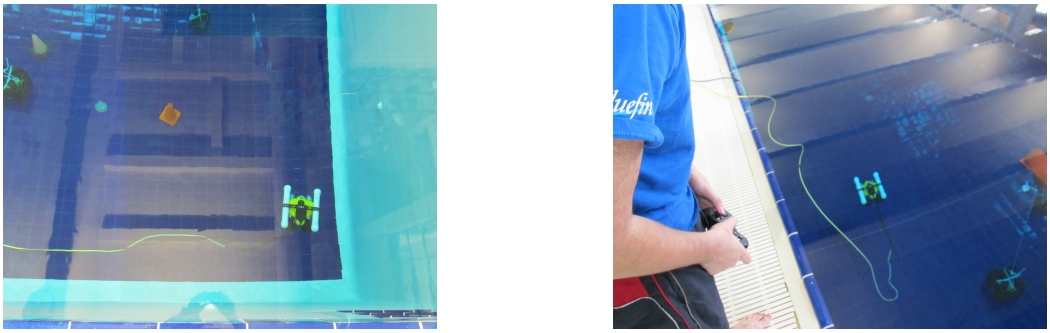
Figure 4: The Agile Close-Quarters Underwater Autonomous System (ACQUAS)

The Agile Close-Quarters Underwater Autonomous System (ACQUAS), shown in Fig. 4a, was developed at the Naval Postgraduate School (NPS) for Close-Quarters Operations (CQOs) with divers [21]. ACQUAS is equipped with a Doppler velocity log (DVL), inertial navigation system (INS), and a GPS, when at the surface, for maintaining position estimates. ACQUAS also has a manipulator for underwater interventions. The manipulator was used to transport scientific samples between Aquanauts during NASA NEEMO Mission 20, as shown in Fig. 4b. Relevant to this work, ACQUAS is equipped with 900 kHz 2D imaging sonars that are oriented in both horizontal and vertical configurations. We were invited by faculty at NPS to collect data of divers with ACQUAS during NASA NEEMO Mission 20.

3.3 Data Collection

3.3.1 Georgia Tech Olympic Dive Well

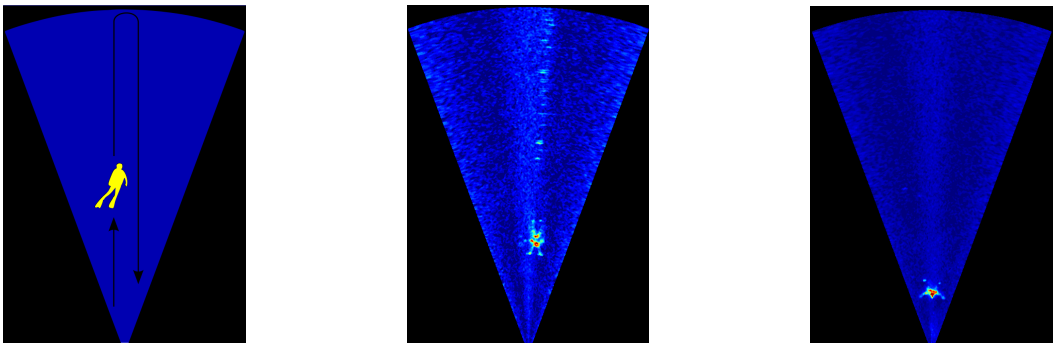
The first time we realized that a human's swimming movements could be detected in 2D imaging sonar data was coincidental. The BlueView P900-45 2D imaging sonar was collecting sonar data of an Autonomous Surface Vehicle (ASV) and when one of the graduate students jumped in the pool to fetch the ASV, the swimmer's movements were clearly visible in the live sonar data. This was the inspiration behind using a 2D imaging sonar to detect and track a diver with an underwater robot. The preliminary results led us to return to the Georgia Tech Dive Pool and conduct several specific data collection scenarios.



(a) The ROV in the Georgia Tech dive pool. (b) An operator controlling the ROV.

Figure 5: Collecting data in the Georgia Tech Dive Pool.

The Georgia Tech Olympic Dive Pool was built in 1996 for the Olympics and is approximately 40 feet in depth. We used the pool to collect 2D imaging sonar data sets of a human swimming in the pool on two separate occasions. The BlueView P900-45 was attached to the side of the pool and submerged approximately 1 m. One of the graduate students in the laboratory was instructed to swim directly away from and then back toward the sonar head, as shown in Fig. 6a, as the sonar data was recorded at a topside laptop. The swimmer was also instructed to remain approximately 2 m underwater during the process. The swimmer's arms and legs are clearly visible in Fig. 6b.



(a) The diver's trajectory during data collection.

(b) Sonar image of a swimmer in a pool.

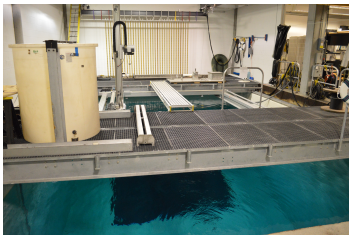
(c) Sonar image of a swimmer in a lake.

Figure 6: Sonar images of the swimmer in a pool and a lake.

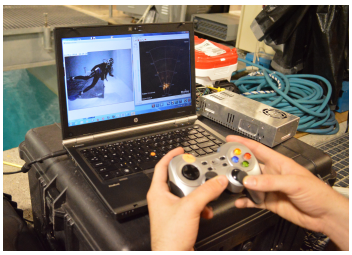
While the swimmer's movements were visible in the pool sonar data, acoustic reflections in the pool created artifacts in the sonar imagery. An additional pool data set was collected in February 2013, in which the diver swam in a trajectory that was perpendicular to the sonar head.

3.3.2 Georgia Tech Acoustic Water Tank

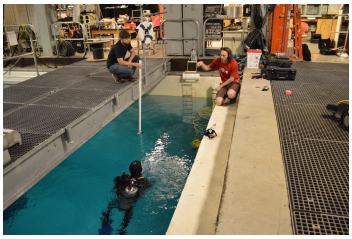
The Georgia Tech Acoustic Tank, shown in Fig. 7a, is located in the Love Building at Georgia Tech and is often used to test underwater communications. During the installation of a mechanical platform in the dive well, we collected 2D imaging sonar data, video from the ROV's camera, and external high-definition video of the diver. Also, several Underwater Human-Robot Interaction (UHRI) scenarios were carried out in which the robot was teleoperated from the surface, as shown in Fig. 7b and Fig. 7c. The scenarios included the diver



(a) The Georgia Tech Acoustic Dive Tank.



(b) The basestation that was used to control and monitor the ROV.



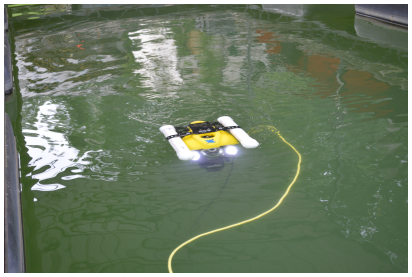
(c) Discussing the next scenario with the diver.

Figure 7: Data collection at the Georgia Tech Acoustic Dive Well.

and robot exchanging tools, navigating in leader-follower formations, and communicating via lights and hand gestures.

3.3.3 Lake Lanier, Georgia

(a) The dock used to launch the ROV.

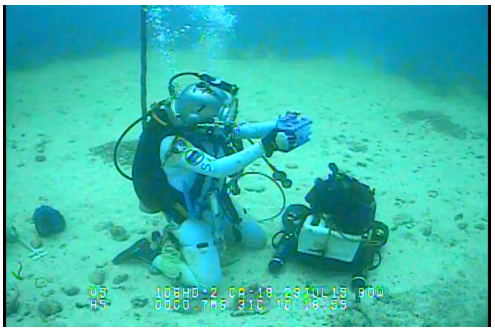


(b) The ROV collecting data in Lake Lanier.

Figure 8: Collecting data in Lake Lanier.

The next data collection session occurred in Lake Lanier, a man-made lake north of Atlanta. The sonar was attached to a dock, which is shown in Fig. 8a, and the swimmer was instructed to swim directly away from and then back toward the sonar head. Opposed to the sonar data collected in the pool, the data collected in the lake contained fewer artifacts, as shown in Fig. 6c. The muddy floor of the lake absorbed acoustic reflections, resulting in a less noisy image. A second data set was collected in October 2012, in which we experimented with ensonifying the swimmer at various vertical-angles relative to the swimmer.

(a) The ACQUAS with the Aquarius habitat's life support buoy.



(b) An aquanaut collecting samples during an EVA.

Figure 9: The ACQUAS and an aquanaut from NASA NEEMO 20.

3.3.4 NASA NEEMO Mission 20

The final data collection scenario took place at the NASA Extreme Environment Mission Operations (NEEMO) Mission 20 at the Aquarius habitat in Key Largo, Florida. The purpose of the NASA NEEMO mission is to provide a convincing analog to space exploration for astronauts-in-training and aquanauts. During the missions, aquanauts experiment with new human-computer interaction tools, experience delayed communications, and conduct extravehicular activity (EVA) missions while wearing environmental suits. An image taken from the ACQUAS of an aquanaut collecting underwater samples during an EVA is shown in Fig. 9. We were invited by a faculty member at the Naval Postgraduate School (NPS) to attend the NASA NEEMO Mission 20 and use the ACQUAS to collect 2D imaging sonar data of the Aquarius habitat, marine life, support scuba divers, and the Aquanauts.

3.3.5 Environmental Conditions Summary

A summary of the environmental conditions that existed during our data collection events is tabulated in Table 2. We were able to record water temperature from the underwater robot's temperature sensor, but turbidity, amount of noise from multipath propagation, and salinity had to be qualitatively estimated. Turbidity refers to the cloudiness of the water as a result of silt, sand, and organic matter suspended in the water column. A highly-turbid environment has low-visibility. Salinity is a measure of how much salt is in the water. The

Table 2: Environmental Conditions During Data Collection Events

Data Collection Event	Turbidity	Temperature	Multipath	Salinity
GT Olympic Dive Well	Low	26°C	High	Low
GT Acoustic Water Tank	Low	25°C	Medium	Low
Lake Lanier, GA	High	29°C	Low	Low
NASA NEEMO 20	Medium	30°C	Medium	High

only salt water condition that we tested in was at NASA NEEMO 20. Most commercial 900 kHz 2D imaging sonars allow the user to configure the estimated speed-of-sound in water due to water salinity and temperature. By ensuring the sonar has the appropriate values for salinity and temperature, distortion in the sonar image can be minimized. However, if the acoustic wave crosses a thermocline, a distinct change in temperature across the water column, acoustic distortion can take place. We also included the relative amounts of multipath propagation noise that we noted during data collection. The sides and bottoms of pools reflect acoustic energy causing ghost images to appear in the sonar data. In natural settings, such as lakes and oceans, there are fewer rigid structures that reflect acoustic energy well, resulting in less multipath noise. The water turbidity could potentially affect the quality of the sonar image, but we never experienced deteriorating sonar images due to water turbidity, even in the highly turbid Lake Lanier. The water turbidity would probably have to possess pieces of debris on the order of centimeters in size before it affected the 2D imaging sonar data.

3.4 Diver Operational Modes

The way in which humans operate underwater is heavily based on the type of equipment they are currently using. Clearly, a human without any artificial breathing devices can only stay submerged for a short period of time and they are limited to shallow depths. On the other end of the spectrum, divers wearing environmental suits, as shown in Fig. 9b, can descend to depths of up to 400 feet for extended periods of time. However, environmental suits also affect how a diver moves underwater. An environmental suit limits a divers ability to "swim." Instead, they have to walk on the ocean floor or man-made structures. Between the complexity of environmental suits and the limited ability of free-diving is self-contained

underwater breathing apparatus (SCUBA) diving. Scuba divers carefully plan their diving missions to minimize the possibility of decompression illness. They do this by specifying the times at which they will remain at each depth during the dive and then use watches and diving computers to track their actual depths and times. In addition to being equipped with tanks and underwater breathing devices, divers use fins on their feet to facilitate swimming underwater and reduce energy usage. During the NASA NEEMO data collection scenarios many of the scuba divers made use of underwater "scooters" that used propellers to pull the diver through the water.

Even though we recorded divers operating in various modes over the course of this research, we decided to focus on detecting and tracking divers in a specific operating mode: scuba diving while swimming with fins. It was decided that detecting divers in every operating mode was too broad and outside the scope for a single dissertation. This mode was chosen for a number of reasons. First, scuba diving is a prolific form of underwater diving. Recreational divers use scuba gear. Also, industrial and military divers are typically required to master scuba-based diving before moving on to environmental suits and rebreathing apparatuses. Thus, being able to detect a diver using scuba equipment would have a widespread effect on a very popular form of diving. Second, divers that wear environmental suits have to walk on the seafloor. Unfortunately, this means that a diver ensonified with a horizontally-oriented 2D imaging sonar can hardly distinguish a diver from a man-made structure. An example of a man-made structure and two divers wearing environmental suits from the NASA NEEMO data collection is shown in Fig. 10. The sonar data was

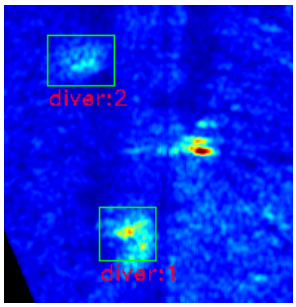


Figure 10: Two divers wearing environmental suits and a man-made structure.

hand-annotated shortly after the mission in order to capture the divers' truth positions.

It is very difficult for the human eye to distinguish between the two diver objects and the man-made structure in the middle of the image. This is opposed to the sonar image of the scuba diver shown in Fig. 11. The image of the scuba diver in Fig. 11 is an ideal image, as

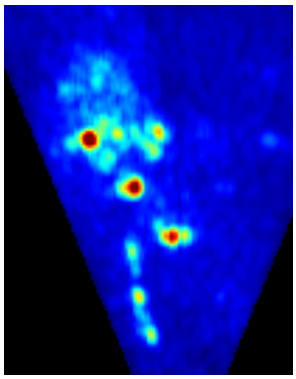


Figure 11: A swimming scuba diver being ensonfied with a 2D imaging sonar.

the diver's abdomen and two legs are clearly visible in this single frame.

Where applicable, we focus our research on the detection and tracking of swimming scuba divers. In the final chapter of this dissertation, we point out several areas of continuing research that involve detecting divers in environmental suits and other modes of operation.

CHAPTER IV

UNDERWATER HUMAN-ROBOT INTERACTION: A CASE STUDY

4.1 Introduction

The purpose of this chapter is to highlight the inspirations for the diver detection and tracking algorithms we develop in the later chapters of this dissertation. We had initially devised these Underwater Human-Robot Interaction experiments to study underwater communication, leader-follower configurations, and tool exchanges. However, we quickly learned that just being able to detect the diver's position was going to be a difficult problem for the Underwater Robotic Assistant (UWRA). Still, the work in this chapter provides insights for how we arrived at focusing on diver detection. Much of the sonar data that was collected during these experiments was used to build and test our diver classifier.

A UWRA could quickly ferry tools or scientific samples to and from the surface; it is dangerous for a diver to quickly surface due to decompression effects on the human body. Also, the UWRA could navigate the diver to the worksite or help find lost divers and lead them back to safety. Similar to how a human dive buddy monitors a buddy for anomalous behaviors to detect potential physiological problems, a UWRA could implement diver activity recognition with its sensors to improve diver safety. Finally, a UWRA could use its built-in lighting system to provide illumination of the diver's workspace in dark and murky environments. While all of these potential applications would surely improve a human diver's efficiency, a major impediment to the realization of these applications is the communication between the diver and the UWRA. The UWRA will have to be able to interpret commands from the diver, such as "Fetch tool X," "Follow me," or "Illuminate this workspace," either through explicit commands from the diver or by interpreting the diver's actions. However, due to physical properties in the underwater domain, traditional methods of communicating information from a human to a robot, such as voice commands, are difficult to achieve. The water medium greatly attenuates radio signals that could

carry voice commands and acoustic communications are prone to dropped packets due to multipath effects. Also, since many of the underwater worksites may take place in low-visibility situations, the UWRA cannot rely upon its optical cameras alone for diver detection and activity recognition. This work focuses on analyzing several proposed methods of underwater communication between a diver and a robot. Some of the communication methods discussed were then used during the installation of an underwater platform in which a human diver was accompanied by a UWRA. This installation process provided a concrete case-study for analyzing what is involved in planning a dive-mission and executing the mission with the assistance of a UWRA. Ideas from Joint Coordination Theory, such as planned and emergent coordination were used to frame the phases in the underwater construction operation. During the mission, a novel technique of using the UWRA's lights to communicate information between the diver and the surface team about mission progress was implemented and tested.

4.2 Underwater Communication

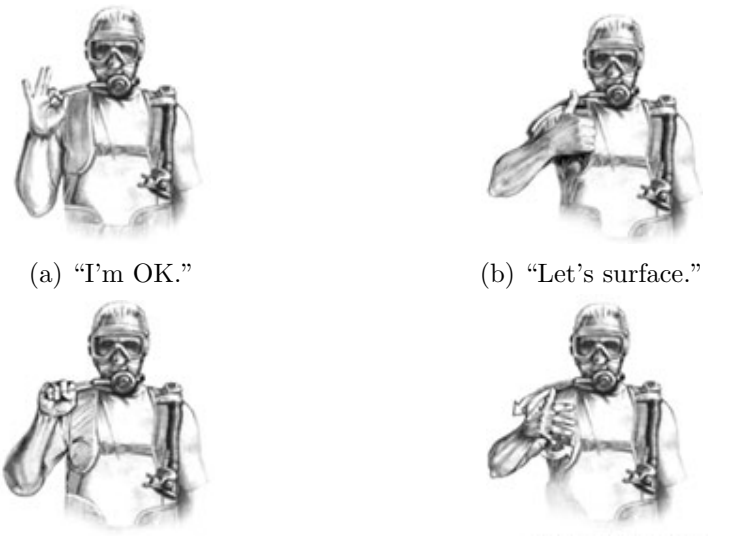
Using robotics in the underwater domain has always been hindered by the difficulty in remotely communicating with an autonomous system. Often, the first step in making a system completely autonomous involves testing the system in a wireless teleoperation mode. Gradually, autonomous features are added to the system. Underwater wireless communication through radio waves is nearly impossible because radio waves are highly attenuated in the water medium, opposed to radio waves in open air [1]. Thus, the primary method of underwater wireless communication involves the use of acoustic communications, which are low-bandwidth and lossy due to the multipath issues of long-wavelength communications. With an underwater robotic system, wireless teleoperation is nearly impossible due to the long time-delay between the operator issuing a control command and the operator receiving sensor data. Thus, the practical use of underwater robotics often involves a tether, over which all communications and sensory data are exchanged. Underwater robots that use a tether are typically referred to as "Remotely Operated Vehicles" (ROV) and have proven

to be extremely useful in underwater missions involving search-and-rescue, salvage, archeology, and inspection. Untethered underwater robots are typically called "Autonomous Underwater Vehicles" (AUV) and require a great deal of autonomy because they can only communicate a few bytes per second over a wireless acoustic communication link. Although not extensive, a number of AUVs have successfully been used in long-range scientific sampling and mapping missions [30]. Given the challenges of underwater communication, in this paper we address the issues of a diver interacting with an underwater robot. It is proposed that the key to solving this problem is to model the solution base on how human divers interact and communicate with each other. While it is obvious that divers use explicit communication, such as hand signals to convey information, divers also communicate through posturing and other implicit forms of communication.

4.2.1 Explicit Underwater Communication

The distinction between explicit and implicit communication can be ambiguous. However, an attempt to categorize explicit communication methods was made based on the assumption that explicit communications involve the use of the diver's hands or tools. Also, while a diver's posture may be implicit because the posture is a consequence of a separate explicit action, an autonomous system that is "programmed" to communicate information through "implicit" methods is really not implicit at all. Based on this concept, we have segmented explicit communications into two categories: "Diver-to-Robot" and "Robot-to-Diver" communication methods. Another distinction between a human's communication methods and a robot's is that until an underwater humanoid robot is developed, the robot's methods of conveying information will inherently be different than the methods that a diver can employ. Still, it is a goal of this research to not greatly alter the standard behavior of the diver, but the diver may have to be trained to understand some of the behaviors of the robot that do not directly translate to diver behaviors. Thus, explicit communications have been segmented into "Diver-to-Robot" and "Robot-to-Diver" communication methods.

Recreational divers are tested on their use of hand signals before they are allowed to dive in open waters. They are required to know hand signals for messages that translate to



(a) "I'm OK." (b) "Let's surface."

(c) "Low on air." (d) "Something is wrong."

Figure 12: Standard Scuba Diving Hand Signals [49].

"I'm OK," "Let's surface," "I'm low on air," "Something is wrong," and other important phrases, as shown in Fig. 12. These are explicit commands that divers use to convey simple phrases[49]. Of course, for a diver to interpret the hand signals from another diver, the two divers must be within visual range, which could vary depending on lighting conditions and the water turbidity. Thus, for a UWRA to interpret hand signals, an appropriate assumption is that the robot and diver are within visual range of each other. This allows the robot to use its optical camera to interpret the diver's hand signals without having to rely on an expensive sonar. Another common method of diver communication is the use of underwater slates to convey information in written form. With an underwater slate, divers can communicate messages that were not defined before the dive; thus, providing a flexibility that is not possible with the basic diving hand signals [49]. It is conceivable that a diver could communicate messages to a UWRA by writing on an underwater slate and then presenting the written message to the UWRA's optical camera. Given that a sufficient optical character recognition (OCR) program exists on the UWRA's computer, the robot could interpret messages from the diver. Thus, the UWRA could operate at the same level of Human-Robot Interaction (HRI) that exists on voice-driven robots, albeit with a slight delay in information transfer, which is a result of the diver having to write down

the message. If completely arbitrary messages are not required to complete the mission, the underwater slate could have many predefined messages printed on it. The diver could merely point to or circle the message he wished to send to the robot while holding the slate in front of the robot's optical camera.

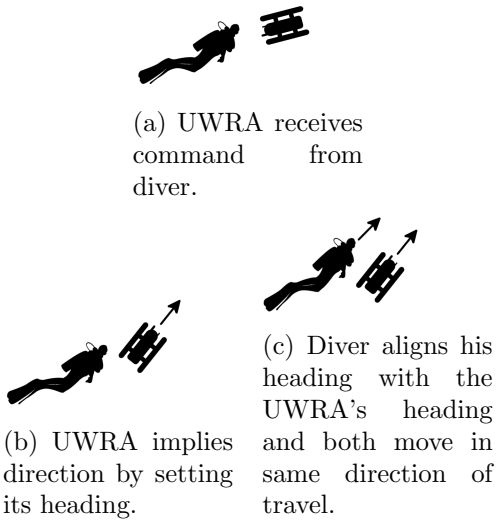
While recreational scuba diving equipment does not lend itself to voice communications because the diver has a regulator in his mouth, many commercial divers wear head units that allow the diver to speak into a microphone [53]. If both the diver and the UWRA were tethered to the surface, the voice communications could be routed from the diver, to the surface, and back down to the UWRA. A form of communication that many cave divers use involves the use of flashlights [46]. Cave divers are usually restricted in their movements due to the tight caves in which they have to navigate. This makes it difficult for divers to use hand signals because they are often head-to-toe while moving through a tight space. Thus, if a following diver needs to acquire the attention of the leading diver, the following diver will rapidly move his flash light, such that the lead diver can see the light [46]. Of course, divers will agree upon what individual light signals signify before the dive, but a rapidly moving light often denotes urgency, while a steadily moving light denotes the need for a calm information exchange. It is conceivable for a UWRA to interpret basic light signals with its optical camera to improve its estimate of the diver's internal state.

The goal in robot-to-diver communication is to make the interaction between human and robot similar to the interaction between two humans. Many ROVs are equipped with high-powered lighting systems to provide illumination in the dark. These same lights could be used to provide feedback to the diver during an interaction. For example, the duration and number of lighting modulations could denote "Acknowledge" or "Not Acknowledge" commands. Also, the lights could be used to attract the diver's attention for an air supply safety check. In a typical interaction, the diver could request a tool from the UWRA using hand signals and the UWRA could either "Acknowledge" or "Not Acknowledge" the hand signals from the diver before starting the task. Unfortunately, since the lights are not a completely natural interaction, the diver would have to be trained to decode the light modulations into messages. As underwater robots are improved and adapted for human

interaction, an obvious choice for a human-computer interface is the digital display. A display, similar to the display used in the Swimoid [61], could convey complex information to the diver.

4.2.2 Implicit Underwater Communication

Human divers use body posture to identify regions of interest or denote a direction of travel. In this work, this form of communication is considered implicit because the diver is not specifically using hands or tools to communicate information. Unfortunately, given its fairly vague definition, there are limitless examples of underwater implicit communication. However, two concrete examples of implicit communication will be discussed. It would be naive to think that the task of leading a diver to a worksite is inherently simple. When two humans engage in leader-follower behavior, the leader does not assume that the follower is always directly behind. The leader occasionally confirms that the follower is nearby [49]. The leader may also choose to swim slightly in front, but to the side of the follower. This is a much safer formation because if the follower expends the air supply, the leader is right beside the follower and can provide the secondary regulator. If the leader was directly in front of the follower, it could take until the end of the next "check in" period before the leader realized that the follower was out of air. When a UWRA leads a human diver to a worksite, these factors must be taken into account. If it is assumed that the UWRA has a sensor suite that enables it to perceive approximately sixty-degrees to the front, the UWRA will have to employ a safe "check in" scheme with the diver. For example, assume that the diver employs one of the previously mentioned communication methods to command the UWRA to lead the diver to a specific worksite, as shown in Fig. 13a. After the UWRA acknowledges the command, it cannot merely start moving in the direction of the goal location without concern for the following diver. However, the leader robot can "imply" the direction of travel by setting its own heading and slowly moving with intent in the direction of travel, as shown in Fig. 13b. This will allow the diver to align the diver's heading with the UWRA's heading and begin transiting towards the next waypoint, as shown in Fig. 13c. Depending on the field-of-view of the UWRA's sensor suite, the UWRA will be able to



(a) UWRA receives command from diver.

(b) UWRA implies direction by setting its heading.

(c) Diver aligns his heading with the UWRA's heading and both move in same direction of travel.

Figure 13: When acting as the "leader" in the navigation scenario, the UWRA will imply direction of travel with its heading.

monitor the diver while both the UWRA and the diver are transiting toward the waypoint, or the UWRA will have to use a probabilistic framework to "check in" with the diver by yawing in the predicted direction of the diver when the uncertainty of the diver's position increases above a threshold. Since the diver will not be directly behind the UWRA, this yawing motion will not greatly inhibit the forward direction of the UWRA. The UWRA can provide assistance in dark settings by illuminating the diver's workspace. However, to accomplish this, the UWRA has to be able to detect the diver's operating area, which may not be completely obvious. In the simple case, the diver will directly face the area-to-be-illuminated and the diver's hands will move within the area of interest. The UWRA will have to take the diver's position into account and maneuver itself in 3D space to provide appropriate lighting.

4.3 Methodology

An experiment was conducted to test the feasibility of using a UWRA with a human diver to complete tasks underwater. Of greatest concern was the feasibility of communicating information between the diver and the robot. Since the autonomous capabilities had not yet been fully specified, this experiment was conducted in the "Wizard-of-Oz" methodology,

in which a human teleoperated the UWRA. Future work will use the results from this experiment to automate the communication.

The following experiment involved a task in which a mechanical platform had to be precisely placed by a diver at the bottom of an underwater acoustic dive well for a set of scientific experiments unrelated to our research. The UWRA was used to monitor the installation process, while the interactions between the diver and the UWRA provided real-world use cases for UHRI.

4.3.1 Mission Planning

Before the diver or the mechanical platform were submerged in the tank, an extensive amount of mission preparation was required. In Joint Coordination Theory, this is referred to as planned coordination since the various roles and tasks were defined before mission execution [34]. Joint Coordination Theory attempts to describe how two or more humans or agents coordinate their actions, to either accomplish a task or to merely perform the same action. The theory makes a specific distinction between two types of coordination: planned and emergent [34]. In planned coordination, agents use the desired outcome and their own concept as a team member to drive their behavior. In emergent coordination, agents coordinate their behavior via perception-action matchings, but the agents do not share joint plans. Most realistic team scenarios consist of a mixture of planned and emergent coordination. First, the mission objectives and installation procedure had to be completely defined and understood by both the diver and the surface team. Second, the required communication signals between the diver and the surface team, via the UWRA, had to be mapped to the objectives and phases of the mission. The team's primary objective was to place a mechanical platform at a specific location at the bottom of an underwater acoustic dive well. The water tank was approximately 10 m in length, 5 m in width, and 5 m in depth. A mission state machine diagram, shown in Fig. 14, was developed to visually describe the order of events and the communication signals between states that would trigger a transition to the next state. The state machine outlines the five main phases of the mission. In the first phase, the surface team, which included the robot, used

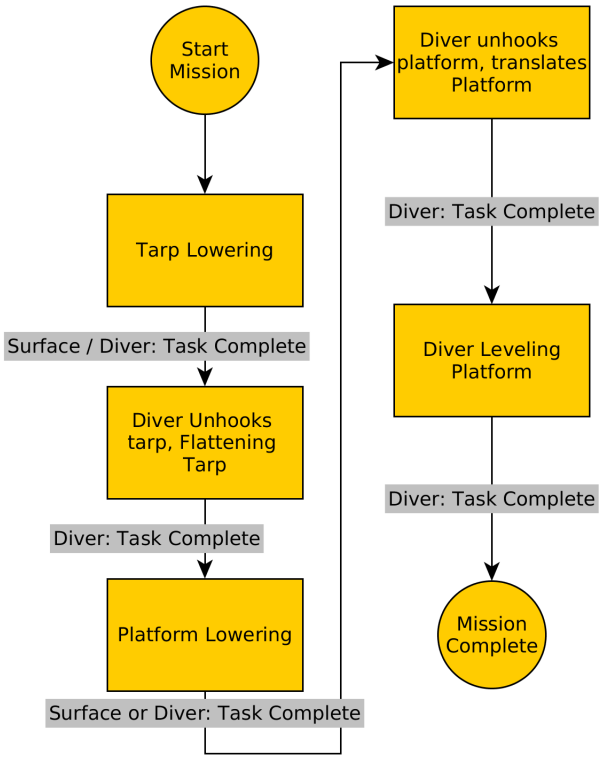


Figure 14: State machine diagram for the mission.

an overhead crane to lower a protective tarp into the tank. After the tarp was completely lowered, the diver unhooked the tarp and proceeded to manually flatten the tarp with his hands. The next step involved the surface team lowering the mechanical platform into the tank. After the platform was completely lowered, the platform was unhooked and translated over to the designated origin in the X-Y plane with the assistance of buoyancy bags. A plumb bob was suspended from the origin in 3D space and the diver used the plumb bob to align the platform with the origin. Since the bottom of the tank was not level, the mechanical platform was outfitted with adjustable feet that could be lowered or raised. Limiting the diver's bottom time was important for both air consumption concerns and due to the fairly cold water conditions. Similar to how human dive buddies agree upon specific hand signals for the upcoming dive, the diver and the surface team agreed upon the definitions of three diver hand signals: "OK," "Something's wrong," and "Task Complete." The hand signals for "OK" and "Something's wrong," which are shown in Fig. 12a and Fig. 12d, respectively, were taken from common scuba diving hand signals. However the

"Task Complete" signal was created for use with the UWRA and consists of the diver making two fists and raising his hands slightly above his shoulders. Communication from the UWRA to the diver was accomplished by modulating the brightness of the UWRA's lights in specific patterns. To communicate the "Not Acknowledge" or "No" command, the UWRA blinked its lights two times within the interval, t_{resp}. In order for the UWRA to communicate the "Acknowledge" or "Yes" command, the UWRA blinked its lights one time within the same time interval, t_{resp}. If the UWRA required the attention of the diver, it would send the "Attention" command by quickly flashing its lights on and off. Finally, the UWRA communicated the "Task Complete" message by increasing the brightness of the lights to full brightness and then decreasing the brightness of the lights until they were completely off. Plots of the brightness of the UWRA's lights versus time are shown in Fig. 15. A transition between mission states occurred when either the surface team with

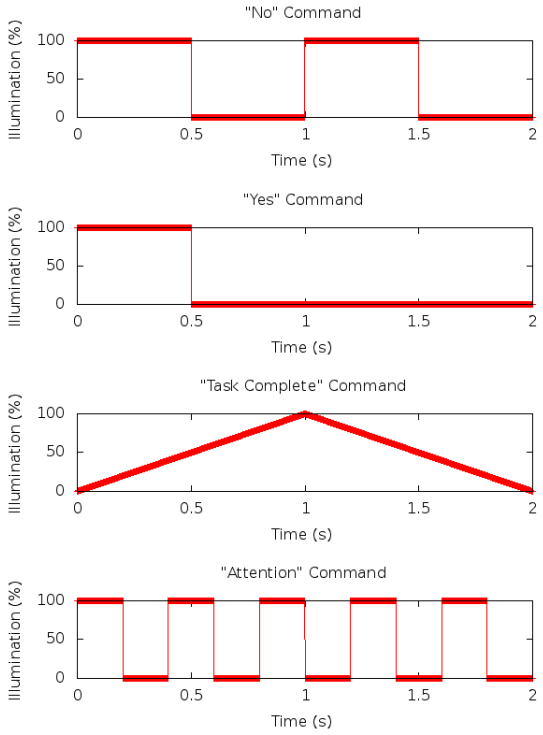


Figure 15: The four UWRA illumination sequences that were used during the field tests to communicate information from the UWRA to the diver. In these plots, $t_{resp} = 2$.

the UWRA or the diver issued a "Task Complete" command. In Fig. 14, each transition arrow is labeled with either "Surface/Diver: Task Complete" or "Diver: Task Complete."

The “Surface/Diver: Task Complete” label meant that either the surface team or the diver could issue a “Task Complete” command; thus, transitioning the entire mission to the next state. The “Diver: Task Complete” label meant that only the diver could issue a “Task Complete” command. If either the surface team or the diver issued the “Task Complete” command, the other entity had to acknowledge the command.

4.3.1.1 RESULTS & DISCUSSION

Besides the fact that the mechanical platform was successfully installed at the bottom of the acoustic tank, most of the results of this experiment were qualitative. The diver was questioned after the installation and he stated that he could decipher the differences between “Acknowledge” and “Not Acknowledge” commands. Also, the light command that was supposed to acquire the attention of the diver worked very well. At one point during the dive, when the diver was adjusting the platform’s level, the “Attention” command was issued to request the diver’s air supply. The diver almost immediately stopped using his wrench and looked directly at the UWRA. An example of the diver and UWRA using light and hand signals to exchange information during the experiment is shown in Fig. 16. One major issue found with using lights to communicate information to a diver is that the

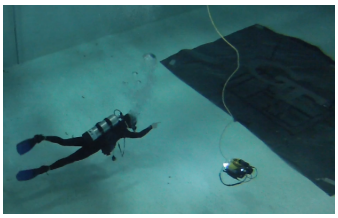


Figure 16: The diver and robot communicating during the experiment.

lights might disorient the diver, especially with missions in dark areas. If the UWRA’s lights are the only light sources available, switching the UWRA lights on and off could possibly confuse the diver. Instead of completely turning off the lights, the UWRA could dim the lights. This experiment also highlighted many of the technical challenges that must be overcome before a fully autonomous robot can assist a diver. For example, the actual robotic platform has to be able to hover in place. Being able to precisely hover in place is a critical capability of a UWRA for it to be able to perform illumination, communication,

and tool hand-off tasks.

In terms of the UWRA's optical camera, it provided the surface team with a horizontal view of the diver that could not otherwise be viewed from the surface. The diver's intentions to either descend or ascend were noticeable by observing the direction in which the diver's face mask was directed. Also, the surface team was able to monitor the diver for strange behaviors and indicators of stress. Since the water was fairly cold, the surface team was able to detect some shivering in the diver while he worked. Unfortunately, the ease in which visual results were obtained was most likely skewed by the fact that the tests were conducted in a filtered indoor acoustic tank. In a realistic situation, the UWRA will have to use its forward-looking sonar to detect the diver at long-range. Still, an activity or intention recognition module could detect the type of tool that the diver is equipped with, detect a limited set of the diver's movement's, and then provide a hypothesis for the diver's current activity. By detecting the diver's activity and state, the UWRA would be reducing the diver's workload by limiting the number of communication exchanges.

An important note about the use of the BlueView 2D Imaging Sonar with a diver is that the sonar was incapable of detecting the diver when placed at the surface, above the diver, as shown in Fig. 17. The bubbles from the diver's exhaled air expanded as they rose to the surface, completely obscuring the sonar's view of the diver. While it was previously thought that the diver could be detected with a forward-looking sonar from the surface, it was shown that the diver's exhaled bubbles greatly obscured the diver's body in the sonar image. Thus, the UWRA's movement will be restricted by the presence of the diver's exhaled bubbles. The UWRA will have to remain close to the same horizontal plane as the diver if the UWRA is to track the diver with its forward-looking sonar. Since the diver was undetectable in sonar data collected at the surface, the sonar was lowered to the same horizontal plane as the diver. As shown in Fig. 18, the diver's legs and upper body were clearly visible in the sonar image when the diver and the sonar were located in the same horizontal plane.

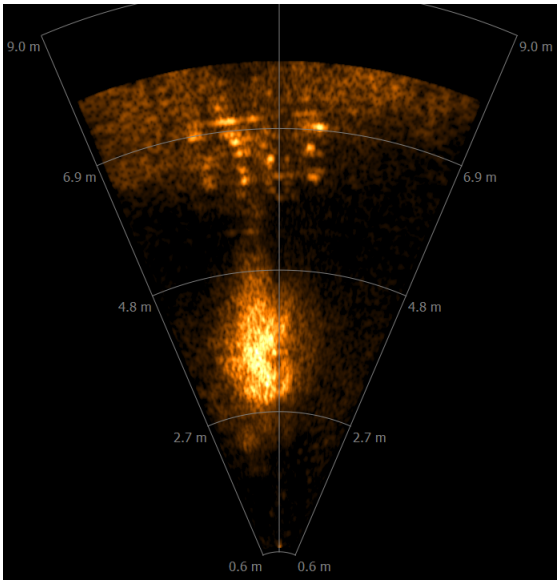


Figure 17: Sonar image of diver taken above from surface. The diver's body and limbs were obscured by the diver's exhaled air bubbles.

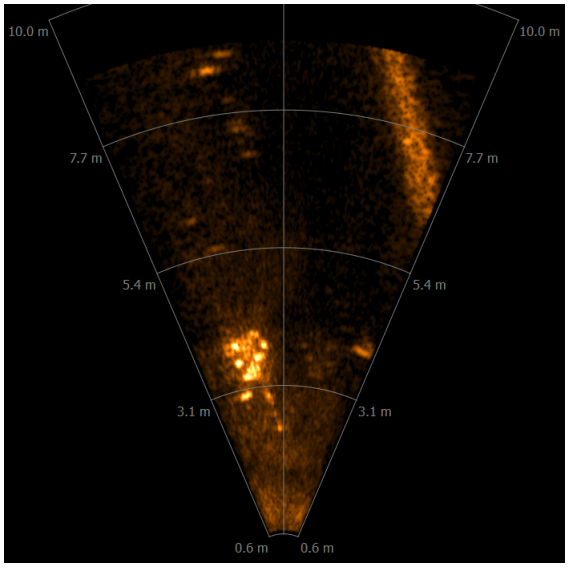


Figure 18: Sonar image of diver taken from diver's horizontal plane as the diver swam away from the sonar head.

4.4 Conclusions

The primary goal of this experiment was to plan and execute an underwater mission in which a human diver and a robot interacted to complete a task. While, the "Wizard-of-Oz" methodology was employed instead of implementing true autonomy on the UWRA, this experiment proved that a robot could assist a diver. There were a number of underwater communication methodologies that were discussed previously and the system of communicating with the diver via the lighting system and simple hand signals was implemented. The surface team and the diver were able to overcome the limited amount of information that could be transmitted with these methods by making an extensive mission plan. This sort of "planned coordination" is not foreign to experienced scuba divers that discuss precisely how many minutes they will remain at each depth before each dive. For even more complex missions, the diver will probably have to be equipped with a waterproof list of mission phases. While hand gestures were used during the mission, pre-discussed hand gestures cannot necessarily be considered "emergent coordination." Emergent coordination refers to coordination activities that naturally arise due to implicit communication. Thus, in this experiment, there was no evidence of emergent coordination. Emergent coordination may be present in future experiments, where the UWRA's ability to lead the diver to different worksites will be tested. When the diver aligns their own heading with the heading of the leader UWRA, the diver should exhibit emergent coordination by creating an ad hoc formation.

This chapter provided a description of a real-world use case for UHRI. We explored issues with human-robot underwater communication, leader-follower configurations, and issues with detecting human scuba divers using underwater sensors. After conducting this UHRI case study, we decided to focus on the problem of detecting and tracking the scuba diver with 2D imaging sonar. This is a very real and immediate problem that is unique to the underwater domain. The problem of detecting and tracking the diver has to be solved before UHRI can actually take place without "Wizard-of-Oz" approaches. Also, by studying the problem of detecting divers with underwater sensors we can investigate the limits that the underwater medium may place on sensor capabilities.

CHAPTER V

2D SONAR IMAGE THRESHOLD ALGORITHMS

Before any meaningful information for underwater applications can be extracted from 2D sonar images, the sonar imagery has to pass through a pipeline of image processing filters. These filters reduce noise in the image and reduce the search space in the image for our later classification stage. This chapter first describes how a 2D sonar image is constructed. Then, the image processing techniques used to threshold the image are presented. Three different threshold algorithms are evaluated and operating points for their thresholds are selected with a K-fold cross-validation framework. One of the threshold algorithms presented is a novel adaptive threshold algorithm that outperforms the other commonly used threshold algorithms.

5.1 2D Sonar Image Construction

A 2D imaging sonar is an active sonar that constructs a sonar image by emitting acoustic signals from its sonar head. When the acoustic waves come into contact with objects in the environment, as shown in Fig. 19, the waves are reflected back towards the sonar head. The

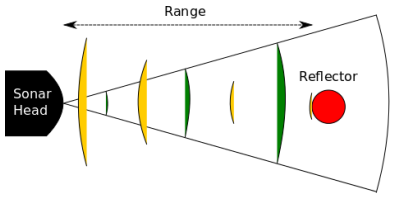


Figure 19: The original sonar pulse that travels from the sonar head to the reflector is shown in green. The reflected sonar pulse that travels from the reflector back to the head is shown in yellow.

range, R, to the reflector is calculated by measuring the time-delay, τ, between the emission of the acoustic wave and the reception. Also, the two-way travel time for the acoustic wave has to be accounted for in the range calculation [29].

$$R = \frac{c\tau}{2} \tag{1}$$

As the acoustic wave propagates through its channel, it experiences spherical spread. Thus, it's intensity, I, experiences an attenuation proportional to R^2.

$$I \propto \frac{1}{R^2} \tag{2}$$

However, if the two-way transmission of the signal is accounted for, the signal's intensity is attenuated by spherical spread on the return trip as well [29].

$$I \propto \frac{1}{R^2}\frac{1}{R^2} = \frac{1}{R^4} \tag{3}$$

In addition to determining the range to a reflector or target and the proportional intensity of the received acoustic echo, another important quantity for the sonar to calculate is the target's relative-bearing. 2D imaging sonar systems determine the target's relative-

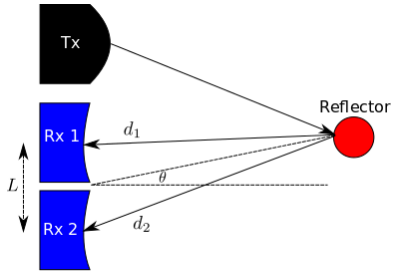


Figure 20: Target's relative-bearing formation.

bearing with an array of acoustic receivers and beamforming circuitry. As shown in Fig. 20, since the two distances, d_1 and d_2, between the reflectors and receivers are different lengths, the reflected acoustic signal will arrive at the two receivers at different times. The time-difference, δt, and the distance between the two receivers, L, can be used to calculate the relative-bearing, θ, to the target.

$$\theta = \sin^{-1}\left(\frac{c\delta t}{L}\right) \tag{4}$$

The same beamforming and relative-bearing estimates can be used to extend to multiple targets.

While 2D imaging sonars can provide an accurate image of the environment, they are plagued by several sources of noise. Both ambient noise in the water column and noise in the electronics of the sonar head contribute to false acoustic returns in the form of salt-and-pepper noise. Also, sonar waves are prone to multi-path reflection, which results in ghosts in the sonar images.

The aggregate sonar geometry of the 2D imaging sonar, the BlueView P900-45, that was used during data collection for this research is shown in Fig. 21. The BlueView P900-45 2D imaging sonar has a field-of-view, θ_S, of 45° and a max range, R_{max}, of 100 m [13]. Shown in Fig. 22, a single sonar beam has an azimuth width, θ_B, of 1° and an elevation width, ϕ_B, of 20°. There are three target-oriented quantities (range, bearing, and acoustic intensity),

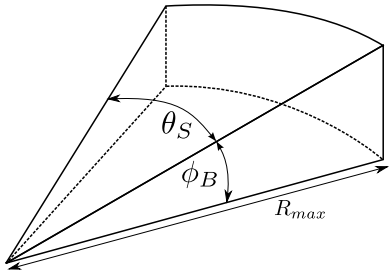


Figure 21: 2D imaging sonar geometry.

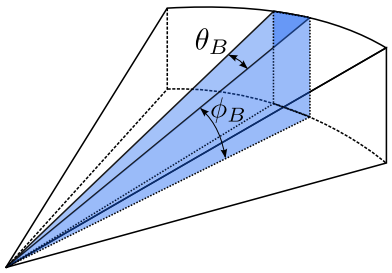


Figure 22: A single sonar beam's geometry.

that can be measured by a 2D imaging sonar. Conveniently, the three measured quantities can be visualized in a two-dimensional gray-scale image, as shown in Fig. 23. The pixel value, p, is proportional to the relative intensity of the received acoustic signal and takes on a gray-scale value, (i.e., $\{p \in \mathbb{Z} \,|\, 0 \leq p \leq 255\}$). Given the range, R, and bearing, θ, to a reflected target, the value for the reflected intensity can be positioned at the proper x and y position with the Polar Coordinate to Cartesian Coordinate transformation provided in

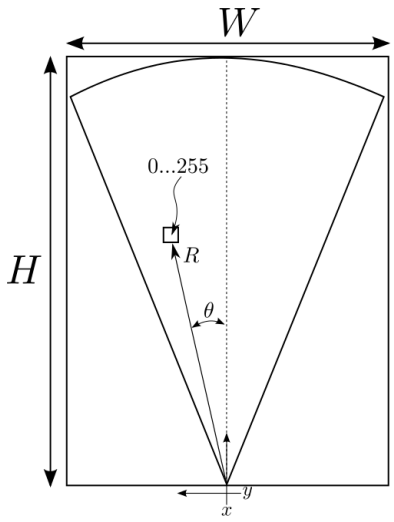


Figure 23: 2D Sonar Image.

eq. (5) and eq. (6).

$$x = R\cos(\theta) \tag{5}$$

$$y = R\sin(\theta) \tag{6}$$

It should be noted that the origins for the two coordinate systems are typically outside of the sonar image. Capturing the target-oriented quantities in a single 2D image enables us to then apply image processing techniques to the problem of detecting and classifying objects in sonar data. Thus, we can use classical image processing techniques to denoise images, state-of-the-art multi-hypothesis trackers to track objects, and machine learning techniques to classify objects.

5.2 Sonar Image Processing Pipeline

The sonar image processing pipeline consists of five primary operations: color conversion, intensity thresholding, data clustering, data association, and object classification. In this section, the color conversion and intensity thresholding operations will be presented. The purpose of the color conversion is to transform the sonar image into a gray-scale image where each pixel's intensity is proportional to the reflected acoustic intensity. The purpose of the thresholding stage is to provide a first-pass detection of possible objects in the scene.

5.2.1 Color Conversion

The pre-processing pipeline uses classical image processing techniques to extract meaningful information from the sonar imagery. While most sonar images carry gray-scale information, sonar data can also be displayed using the Jet color map. A Jet color-map provides a unique mixture of red, green, and blue values to represent a gray-scale value. Thus, the first step in the image pre-processing pipeline is the conversion from the Jet color-map to gray-scale. Each pixel in a color image is represented by a 3-dimensional color vector, $\boldsymbol{C} = [\mathrm{v}_b, \mathrm{v}_g, \mathrm{v}_r]^T$, where each element in the vector, v_i, can take on a value between 0 and 255 (i.e., $\{\mathrm{v}_i \in \mathbb{Z} \,|\, 0 \leq \mathrm{v}_i \leq 255\}$). v_b refers to the blue intensity in the pixel, v_g refers to the green intensity, and v_r refers to the red intensity. The relationship between the gray-scale representation and the Jet color-map representation is shown in Fig. 24. A

Figure 24: Gray-scale (top) and Jet (bottom) color-maps.

value of zero is represented by black in the gray-scale representation and by dark blue in the Jet representation. A value of 255 is represented by white in the gray-scale image and by dark red in the Jet representation. The proportions of red, blue, and green intensities that are used to convert a gray-scale pixel to a Jet color-mapped pixel are shown in Fig. 25. The conversion from a Jet-colored image to a gray-scale image requires a pixel-wise non-linear transformation, $T : \boldsymbol{C} \mapsto \mathrm{g}$, that transforms each color vector into a gray-scale pixel. By dividing the x-axis in Fig. 25 into five distinct regions, we can construct a non-linear transformation that consumes a colored-pixel and generates a gray-scale value. The divisions are denoted by dashed lines. For example, if we assume that $V_{max} = 255$ and $V_{min} = 0$, the first region in Fig. 25 can be defined where $\mathrm{v}_g = V_{min}$ and $\mathrm{v}_r = V_{min}$. This

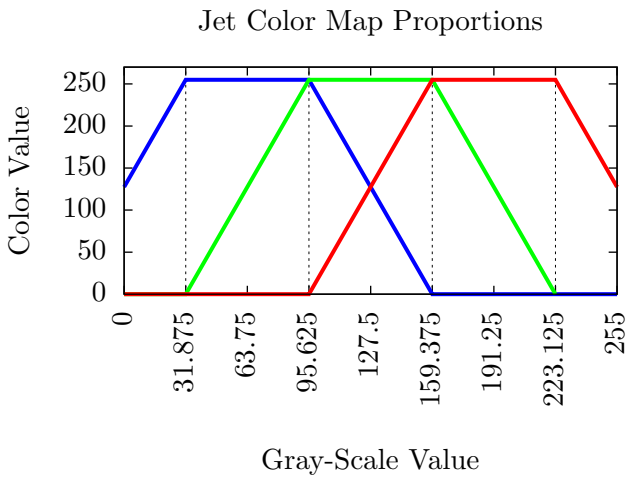


Figure 25: Jet Color Map Proportions.

correlates to the region along the x-axis where the gray-scale value varies from 0 to 31.875. In the second region, the slope, m, of the green line is defined by

$$m = \frac{\Delta y}{\Delta x} \tag{7}$$

$$\frac{\Delta y}{\Delta x} = \frac{V_{max} - V_{min}}{\frac{3}{8}(V_{max} - V_{min}) - \frac{1}{8}(V_{max} - V_{min})} \tag{8}$$

which reduces to

$$m = \frac{V_{max} - V_{min}}{\frac{2}{8}(V_{max} - V_{min})} \tag{9}$$

$$m = 4 \tag{10}$$

The increasing and decreasing lines in Fig. 25 either have the same slope as the green line in the second region or have a negated slope. Using this slope and dividing the plot into five regions based on color values being saturated to either V_{max} or V_{min}, a transformation, shown in eq. (11), can be constructed that converts the Jet's color values to gray-scale values.

$$\text{JetToGray}(\boldsymbol{C}) = \begin{cases} \frac{\mathrm{v}_b - \frac{1}{2}(V_{max} - V_{min})}{4}, & \text{if } \mathrm{v}_g = V_{min} \text{ and } \mathrm{v}_r = V_{min} \\ \frac{\mathrm{v}_g + \frac{1}{2}(V_{max} - V_{min}) - V_{min}}{4}, & \text{if } \mathrm{v}_b = V_{max} \text{ and } \mathrm{v}_r = V_{min} \\ \frac{\mathrm{v}_r + \frac{3}{2}(V_{max} - V_{min}) - V_{min}}{4}, & \text{if } \mathrm{v}_g = V_{max} \\ -\frac{\mathrm{v}_g - \frac{7}{2}(V_{max} - V_{min}) - V_{min}}{4}, & \text{if } \mathrm{v}_b = V_{min} \text{ and } \mathrm{v}_r = V_{max} \\ -\frac{\mathrm{v}_r - \frac{9}{2}V_{max} + \frac{1}{2}V_{min}}{4}, & \text{if } \mathrm{v}_g = V_{min} \text{ and } \mathrm{v}_b = V_{min} \end{cases} \tag{11}$$

If this transformation is applied to an entire Jet-colored image by scanning over the entire image, its equivalent gray-scale image can be generated. A code-listing for the image scanning routine is shown in Listing 5.1.

```
1  for (int r = 0; r < jet_img.rows; r++)
2    for (int c = 0; c < jet_img.cols; c++)
3      gray_image(r,c) = JetToGray(jet_img(r,c));
```

Listing 5.1: Jet to gray-scale conversion.

5.2.2 Data Thresholding

While it is conceptually one of the simplest image processing techniques, data thresholding is an important step in the sonar image processing pipeline. Data thresholding is not typically used when processing optical camera data because the relative intensity of a pixel is not necessarily a consistent feature that can be tracked frame-to-frame. However, a pixel's intensity in a 2D sonar image is directly proportional to the received acoustic amplitude. Thus, a pixel with a high-intensity relative to other pixels is a strong indicator of a valid object in the sonar's field-of-view. This leads to the question: how do we specify a threshold level, l_{th}, that can consistently separate valid objects from noise in the sonar image? If the threshold is set too high (i.e., $l_{th} \approx 255$), we run the risk of filtering out objects that we wish to track, such as divers. However, if the threshold is set too low (i.e., $l_{th} \approx 0$), our threshold filter will allow too many pixels to pass through. Thus, the tracking filters in the later stages may be computationally encumbered by the large number of targets. When processing gray-scale images, the threshold level belongs to the integer set between 0 and 255, but the threshold level should be set such that objects that we want to track are passed, but noise and clutter are rejected. Fortunately, we can use the sonar data sets we collected to select the threshold level. While we have access to sonar data sets, we will also want to know how well the threshold assignment will affect the processing of future data sets. To provide an estimate of the thresholding scheme's ability to handle "unseen" data sets, 10% of the sonar data sets will be designated as the final test set. This final test set will be set

aside and not used during any training or cross-validation phases.

5.2.2.1 Static Threshold

There are three types of "threshold" algorithms that will be compared: the traditional static threshold, a gradient-based threshold, and a novel adaptive threshold. The static threshold filter, shown in eq. (12), is used as a baseline procedure for separating valid objects from sonar noise and clutter.

$$\text{StaticThreshold}(x, y, l_{th}) = \begin{cases} src(x, y), & \text{if } src(x, y) > l_{th} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

In eq. (12), src represents the source image. x and y represent the column and row for each pixel, respectively. Throughout this work we will follow the convention that is found in image processing research; the origin is located at the top-left corner of the image. Thus, x increase from left to right and y increases from top to bottom. The static threshold passes through the source image's pixel if the pixel has a value greater than l_{th}; otherwise, the pixel's value is clamped to zero. However, there are other schemes that provide functionality analogous to the static threshold.

5.2.2.2 Gradient Threshold

The 2D gradient of the sonar image can be computed and then a threshold filter can be applied to the gradient's magnitude [32]. The image gradient is computed by convolving the image, I, with the Sobel derivatives in the x and y directions. The definitions of the image gradient in the x-direction, G_x, and the image gradient in the y-direction, G_y, are provided in eq. (13) and eq. (14) [65].

$$G_x = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \tag{13}$$

$$G_y = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{14}$$

The final image gradient magnitude, G, is computed in eq. (15).

$$G = \sqrt{G_x^2 + G_y^2} \tag{15}$$

Finally, the static threshold filter can be applied to the image gradient's magnitude. Using this gradient threshold method, features in the image that have a strong localized change in pixel intensity are passed through to the next stage of the sonar image processing pipeline.

5.2.2.3 Adaptive Threshold

A third technique involves modifying the threshold level in every frame in order to maintain a fixed ratio between the number of non-zero-valued pixels and the total number of pixels in the image. The ratio, Q, is computed in eq. (16) by summing the number of non-zero pixels in the image, I, and dividing the summation by the total number of pixels in the image.

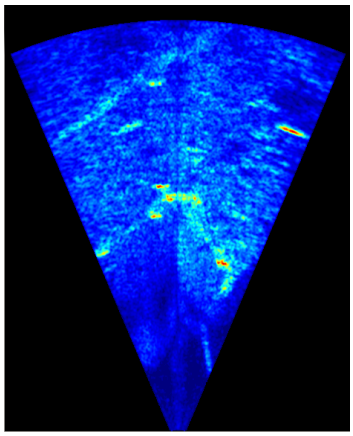
$$Q(I) = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C} \text{sat}(I(i,j))}{\sum_{i=1}^{R}\sum_{j=1}^{C} 1} \tag{16}$$

R is the number of rows in the image and C is the number of columns in the image. The saturation function is defined in eq. (17).

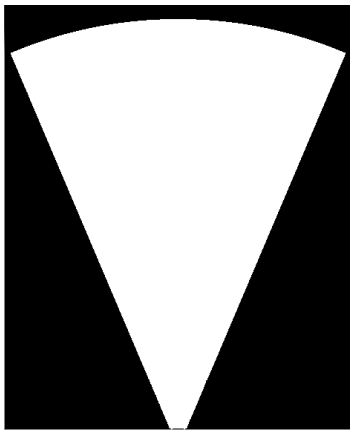
$$\text{sat}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

A unique issue with computing the total number of pixels in a sonar image is that there are a number of pixels in the range-bearing version of the sonar image that will never be non-zero, as shown by the black pixels in Fig. 26a. This is a consequence of the sonar hardware's maximum and minimum bearing angles and the currently specified software-defined maximum range. We can exploit the fact that black is an undefined color vector in the Jet color-map to construct an image mask to define the valid range-bearing pixel locations in the sonar image. The mask operator, δ_m, provided in eq. (18), converts a colored pixel into a binary pixel based on the presence or absence of non-zero color values.

$$\delta_m(\boldsymbol{C}) = \begin{cases} 0, & \text{if } \mathrm{v}_b = 0 \text{ and } \mathrm{v}_g = 0 \text{ and } \mathrm{v}_r = 0 \\ 1, & \text{otherwise} \end{cases} \tag{18}$$



(a) 2D sonar image with Jet color-map.



(b) Binary mask of the sonar image. White denotes valid pixel locations.

Figure 26: Mask generation from the 2D sonar image.

Applying δ_m to every pixel in Fig. 26a results in the binary mask shown in Fig. 26b, where the sonar mask was multiplied by 255 to improve the visualization of the mask. Now that we have a mask that is non-zero at valid pixel locations, we can compute Q by iterating over the mask's domain, D, instead of all rows and columns. The mask's domain is specified in eq. 19.

$$D = \{i \in 1..R,\, j \in 1..C \,|\, \delta_m(i,j) > 0\} \tag{19}$$

Thus, the ratio of the number of non-zero pixels to the number of total pixels is defined over D.

$$Q(I) = \frac{\sum^{D} \text{sat}(I(i,j))}{\sum^{D} 1} \tag{20}$$

While Q could have been computed by ignoring the invalid pixel locations, this computation would have artificially weighted the denominator in eq. (16) such that the numerator could never be equal to the denominator. Thus, by limiting the ratio computation to using only the pixels within the mask's domain, the final value of Q will be less likely to encounter floating point precision issues.

A simplified block diagram for this adaptive threshold procedure is shown in Fig. 27. The algorithm attempts to achieve a desired ratio of number of non-zero valued pixels to total number of pixels by iteratively applying a threshold level, l_{th}, to the gray-scale sonar image and then computing the ratio error, Q_e. The ratio error is used to update the value of

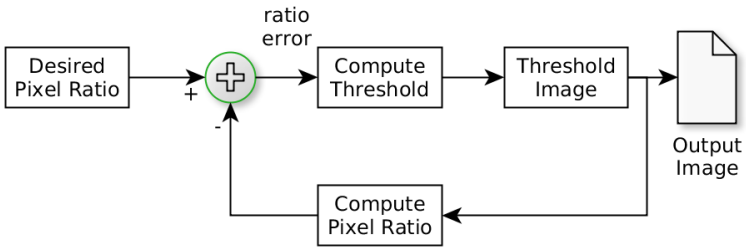


Figure 27: Adaptive threshold routine.

the threshold level for the next iteration. A more detailed outline of the adaptive threshold algorithm is provided in Algorithm 1. The AdaptiveThreshold function is called once for

Algorithm 1 Computation of the Adaptive Threshold Image

Precondition: I_{src} is a gray-scale sonar image
Precondition: $\{Q_d \in \mathbb{R} \,|\, 0 \leq Q_d \leq 1\}$
Precondition: $\{l_{th} \in \mathbb{Z} \,|\, 0 \leq l_{th} \leq 255\}$

```
1:  function AdaptiveThreshold(I_src, Q_d, l_th, i_max)
2:      i ← 0
3:      repeat
4:          I_th ← StaticThreshold(I_src, l_th)        ▷ I_th : Thresholded image
5:          Q_th ← Q(I_th)                              ▷ Q_th : calculated ratio
6:          Q_e ← Q_th − Q_d                            ▷ Q_e : ratio error
7:          l_th ← HQ_e                                 ▷ adjust threshold value
8:          if l_th < 0 then
9:              l_th ← 0
10:         else if l_th > 255 then
11:             l_th ← 255
12:         end if
13:         i ← i + 1
14:     until i > i_max or |Q_e| < ε
15:     return I_th
16: end function
```

each received sonar image, but it iteratively calls the StaticThreshold function. In order to achieve the desired ratio, Q_d, the threshold level is adjusted at Line 7 based on the ratio error, Q_e, and a gain, H. To ensure that l_{th} does not leave the valid range, if-else statements are used to saturate l_{th} between 0 and 255. The AdaptiveThreshold function stops iterating when one of two conditions is met at Line 14: the iteration counter, i, exceeds i_{max}, or when the absolute value of the ratio error is less than a small value, ϵ. i_{max} has to be tuned by the system designer to account for the trade-offs between time-delays in processing each

frame and delays in converging to the threshold level appropriate for the ratio. Since Q_e is a double value, it's equality to zero cannot be robustly computed due to possible precision issues. In this implementation, $\epsilon = 0.00001$. An important implementation note is that the program that calls the AdaptiveThreshold function maintains a value for l_{th} across multiple frames since the value for l_{th} that produced a desirable Q in the previous frame should be used as a starting point for the next frame's threshold value. In this way, the AdaptiveThreshold can quickly achieve the desired ratio in fewer iterations.

5.3 Receiver Operating Characteristic (ROC) Analysis of Threshold Algorithms

The purpose of the threshold algorithms is to pass pixels that belong to the object-of-interest (e.g., a diver) on to the next stage in the image processing pipeline and filter out pixels that do not belong to objects-of-interest. However, each of the three threshold algorithms do not have a single operating point that can be used to assess and compare the algorithms. Instead, each threshold algorithm has a range of operating points that affects the probability that a pixel belonging to an object-of-interest will be passed through the filter or if it will be blocked. Similarly, the threshold operating point affects the probability that a pixel originating from a non-object-of-interest will be passed through the threshold filter. Receiver Operating Characteristic (ROC) plots have been used in binary classification problems to identify an algorithm's operating point and to assess how the operating point will affect the filter's future performance [24]. A ROC plot for a binary classification algorithm is generated by sweeping a single parameter for the algorithm over its valid range and then calculating the True Positive Rate (TPR) and False Positive Rate (FPR) for each operating point. The TPR and FPR are specified in eq. (21) and eq.(22), respectively.

$$TPR = \frac{TP}{TP + FN} \tag{21}$$

$$FPR = \frac{FP}{TN + FP} \tag{22}$$

TP denotes true positives, FN denotes false negatives, TN denotes true negatives, and FP denotes false positives. After the "threshold" parameter for the algorithm is swept across relevant values and the TPR and FPR are calculated for each threshold value, a ROC plot

can be constructed. An example ROC plot is shown in Fig. 28. In this example ROC plot,

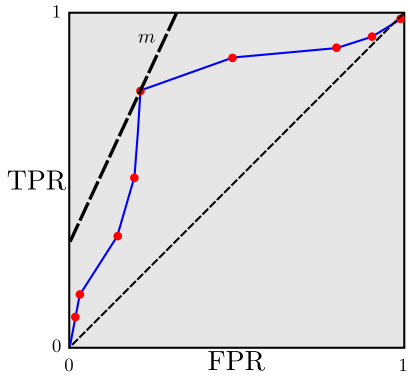


Figure 28: Operating point selection.

each operating point, (FPR_i, TPR_i), is marked with a red circle. The 45° dotted line that extends from the $(0, 0)$ to $(1, 1)$ represents the line of random guessing when the classification algorithm has a 50% chance of correctly identifying the sample. Thus, all operating points to the right of the dotted line are worse than random guessing and operating points to the left of the dotted line are better than random guessing. Once the ROC plot has been constructed, the operating point on the ROC plot has to be selected in order to use the classification algorithm on future unlabeled samples. The method of selecting the operating point consists of first constructing a line that intercepts the y-axis at $(0, 1)$ and with slope, m, based on the number of positive samples, P, the number of negative samples, N, and the costs associated with classifications and misclassifications, as provided in eq. (23) [66].

$$m = \frac{Cost(FP) - Cost(TN)}{Cost(FN) - Cost(TP)} \frac{N}{P} \tag{23}$$

The line is then shifted down and to the right until it intercepts with one of the calculated operating points, which becomes the selected operating point. In Fig. 28, the darker dotted line represents the line with slope, m, that was iteratively moved until it intercepted with an operating point.

In the final evaluation of unseen data sets, there are six statistical measures that will be used to compare the three threshold algorithms: FPR, TPR, *accuracy*, TNR, PPV, and F_1-score. The calculation of the *accuracy* is provided in eq. 24.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{24}$$

The true negative rate (TNR), which is also known as the specificity, is provided in eq. (25) [23].

$$TNR = \frac{TN}{FP + TN} \tag{25}$$

The calculation for the positive predictive value, PPV, which is also known as precision, is provided in eq. (26) [23].

$$PPV = \frac{TP}{TP + FP} \tag{26}$$

Finally, the F_1-score, which is a measure of both the classifier's precision and recall is provided in eq. (27) [55].

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{27}$$

5.3.1 Ground Truth and Annotated Data

In order to count instances of TPs, FPs, TNs, and FNs, a ground truth has to be established. One way of establishing a ground truth automatically is to attach an acoustic beacon directly to the diver that we want to track and establish a Long Baseline (LBL) acoustic network [44]. LBL acoustic networks localize nodes in the network by measuring the two way travel time (TWTT) of acoustic messages. The TWTTs are used to estimate ranges, which allows a node to triangulate itself with respect to other nodes in the network. However, a major drawback to this approach is the cost of developing and setting up the LBL network. Also, attaching an acoustic beacon to the aquanauts would have been a difficult process due to safety regulations. Thus, a more straightforward, but time-consuming approach to obtaining ground truth data of the diver's position was to hand-annotate the sonar images.

For the process of annotating the sonar images, we developed a custom graphical user interface (GUI) that was capable of opening the proprietary sonar image format. A screen shot of the annotation GUI is provided in Fig. 29. The GUI allows the user to place bounding boxes around multiple objects-of-interest in the image and provide an object type and object ID for each bounding box. In Fig. 29, the user placed bounding boxes around two different divers: one diver labeled "diver:1" and the other diver labeled "diver:2." After labeling each frame, the user can save the annotated data in an XML file format for later playback, editing, or classifier assessment.

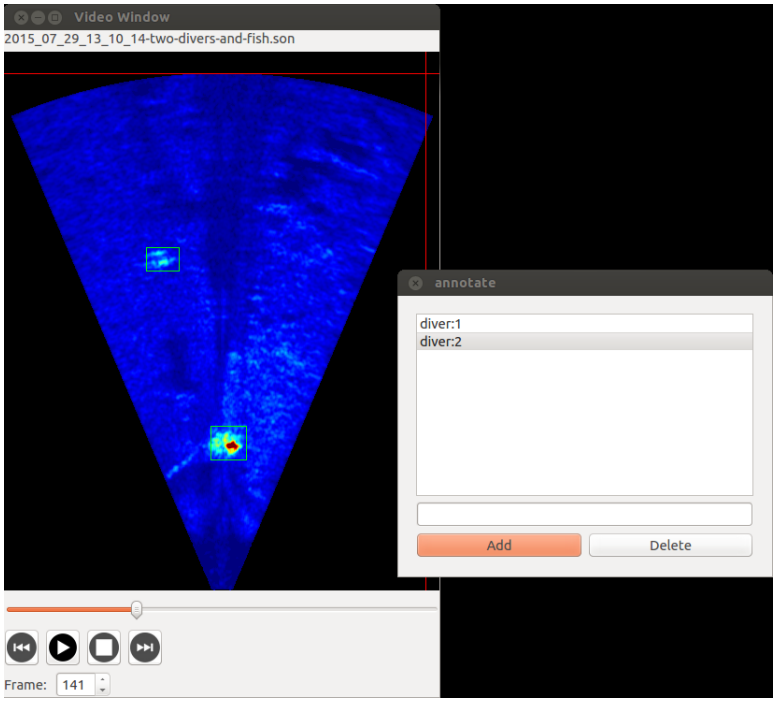


Figure 29: 2D Sonar Image Annotation GUI.

5.3.2 Definition of True Positives, False Positives, True Negatives, and False Negatives

The concept of counting instances of TPs, FPs, TNs, and FNs is intuitive for binary classification schemes where each data sample is either a straightforward positive or a negative sample. However, in our case, we not only want to know if a sonar frame contains a diver, but we also want to know the specific location of the diver in the sonar image. Thus, the definition of a TP only makes sense if the detector declares a positive classification within the bounding box of an annotated diver-object. To determine whether a point is within a rectangle's bounding box, the point's x and y coordinates can be compared against the rectangle's maximum and minimum coordinates. The function, *contains*, in eq. (28), returns *true* if the point, p, is within the rectangle, *rect*, otherwise, it returns $false$.

$$contains(rect, p) = \begin{cases} true, & \text{if } x_{min} < x < x_{max} \text{ and } y_{min} < y < y_{max} \\ false, & \text{otherwise} \end{cases} \tag{28}$$

Assuming standard image coordinates, where the origin is located at the top-left of the image, the x-axis increases with increasing column number, and the y-axis increases with

increasing row number, the rectangle's top-left corner is located at (x_{min}, y_{min}) and the bottom-right corner is located at (x_{max}, y_{max}).

Since the algorithms that are being assessed in this section are pixel-based threshold algorithms, whether a TP, FP, TN, or FN is declared is based on where the pixel is located and whether the pixel in question is zero-valued or non-zero-valued. A first attempt at defining the type of declaration is to declare all non-zero pixels inside of an annotated bounding box as TPs, all non-zero pixels outside of a bounding box as FPs, all zero pixels inside of a bounding box as FNs, and all zero pixels outside of bounding boxes as TNs. However, this scheme results in a disproportionate number of TNs, which skews the FPR to a small value. Also, it might not be appropriate to count FNs for zero-valued pixels inside bounding boxes when there exist other pixels inside of that same bounding box that are non-zero. Perhaps the shape of the object-of-interest was such that annotating the object with a rectangle resulted in a disproportionate number of zero-valued pixels. As such, we need a procedure for counting instance types that scores the threshold algorithms based on the fact that we desire at least one non-zero pixel within the bounding box of objects-of-interest and that we desire zero-valued pixels outside of the positively-labeled bounding boxes. Also, we need a method for balancing the number of positive and negative samples in each sonar image, so that choosing the operating point on the ROC curve by constructing a line of slope, m, as in eq. (23), is not hindered. If the total number of negative samples, N, is much larger than the number of positive samples, P, then m approaches an undefined slope. Also, if P is much larger than N, then m approaches zero.

To achieve our objective of properly counting TPs, we adopt the method of only counting one TP or FN for each hand-annotated positive object. Thus, after the threshold algorithm has been applied to a sonar image, each object's bounding box is scanned for non-zero-valued pixels. If a single non-zero-valued pixel is found within the bounding box, a single TP is declared. The declaration of a single TP is shown in Fig. 30 where non-zero-valued pixels are found within the bounding box of the object labeled "diver:1." If only zero-valued pixels are found in the positive-sample bounding box, a single FN is declared. A single FN is shown in Fig. 30 because there are not any non-zero-valued pixels found within the bounding box

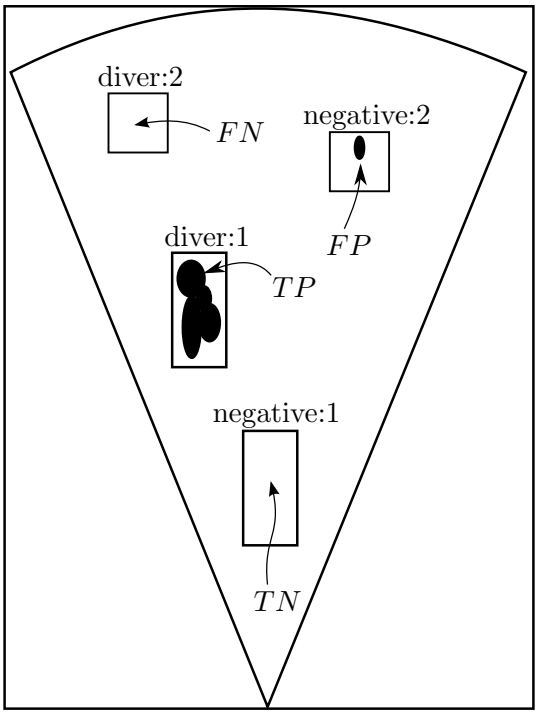


Figure 30: Examples of TP, FP, TN, and FN.

of the hand-annotated "diver:2" object. With this method, there can only be as many TPs or FNs in each sonar image as the number of hand annotated objects-of-interest.

In order to meet our second objective of balancing the number of positive and negative samples to facilitate with the calculation of the slope of the line that selects the operating point, the number of negative samples has to be controlled. This is accomplished by generating negative samples that have bounding boxes of the same sizes as the positive samples' bounding boxes. The generated negative samples are labeled in Fig. 30 as "negative:1" and "negative:2" and are associated with "diver:1" and "diver:2," respectively. The negative samples are not hand-annotated, but are automatically generated based on a number of constraints. The number of negative samples that are generated in each frame is based on a predefined desired ratio, r_d, of number of negative samples to number of positive samples. Thus, if P_i is the number of positive samples in sonar frame i, the number of generated negative samples, N_i, in frame i will be:

$$N_i = r_d P_i. \tag{29}$$

Thus, in Fig. 30, where $r_d = 1$, since there are two positively-labeled samples, two negative

samples are generated. While the sizes of the bounding boxes of the generated negative samples are the same sizes as the bounding boxes of their positive sample counterparts, the positions of the top-left corners of the negative samples are determined by sampling from a uniform distribution, $\mathcal{U}(a, b)$. $\mathcal{U}(a, b)$ is uniformly distributed from a to b, inclusively. For a single negative sample with a bounding box of width, w, and height, h, the coordinate of the top-left corner, (x_{tl}, y_{tl}), of the bounding box is determined by:

$$x_{tl} = \mathcal{U}(0, C - 1 - w) \tag{30}$$

$$y_{tl} = \mathcal{U}(0, R - 1 - h), \tag{31}$$

where C is the number of columns and R is the number rows in the image. By limiting the position of the generated rectangle, we ensure that the rectangle falls within the image's confines. However, it is also important for the generated rectangle to only encompass valid pixels in the sonar image. Thus, after the position of the rectangle is sampled, if any of the pixels inside of the rectangle do not overlap with the sonar's binary mask, the rectangle is discarded. Also, checks ensure that the generated rectangle does not overlap with the bounding boxes of any of the positive samples. Finally, if the generated rectangle does not overlap with any of the previously generated negative rectangles in this frame, the generated rectangle is accepted as a valid negative sample. In total, a newly generated rectangle must undergo four main checks:

1. Does the sonar image completely contain the rectangle?
2. Does the sonar mask completely contain the rectangle?
3. Is the intersection of the rectangle with each of the positive samples the empty set?
4. Is the intersection of the rectangle with each of the previously generated negative samples the empty set?

For each generated rectangle, if the answer to any of these questions is "No," the rectangle's position is resampled until it passes all four of the tests. An example of using a desired ratio of three, $r_d = 3$, to generate negative samples is shown in Fig. 31. The hand-annotated

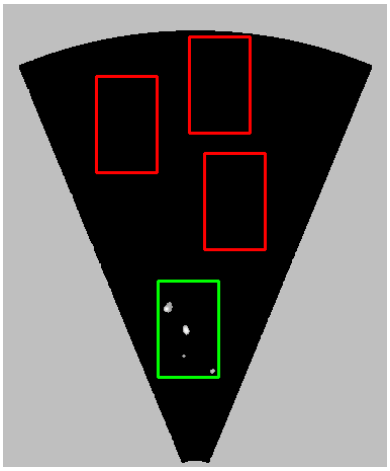


Figure 31: Negative sample generation. The hand annotated rectangle is shown in green and the three generated negative samples are shown in red.

rectangle is shown in green and the three generated negative samples are shown in red. Note that the three generated rectangles are within the mask and do not overlap with any other rectangles in the frame. Depending on r_d, the number of positive samples, and the sizes of the positive samples' bounding boxes, achieving the desired number of generated rectangles could be unlikely. Thus, the number of attempts for resampling the position of the rectangle in each frame is limited to 500.

After generating the valid negative rectangle samples, the rectangles are scanned for non-zero-valued pixels. If a generated rectangle contains any non-zero-valued pixels, a FP is declared for that rectangle. Likewise, if a generated rectangle does not contain any non-zero-valued pixels, a TN is declared for that rectangle. The process is repeated for all generated negative rectangles.

5.4 Cross-validation and Testing Framework

Our objective in this section is to determine the "best" operating point for each of the threshold-based algorithms and then compare the performance of the three algorithms. We already discussed how to generate ROC plots for each of the threshold algorithms and how to select the operating point from the ROC plot in section 5.3. However, the same data that was used to generate the ROC plot and extract the operating point cannot be used for the final evaluation of the classifier [3]. This would lead to an overly optimistic classifier

accuracy. Thus, the sonar data was divided into three subcategories: training data, cross-validation data, and test data. The first step involved appending all sonar frames into a logical vector. This process is shown in Fig. 32, where three independent sonar data files, A, B, and C are appended. Then, 10% of the appended sonar data frames are randomly

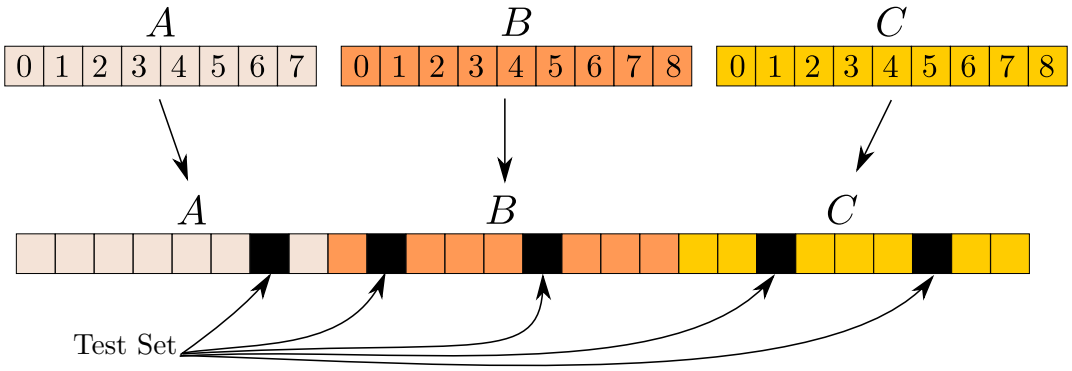


Figure 32: Test Set Selection

selected with a uniform distribution. These randomly selected sonar frames are labeled as the "test set," which is denoted in Fig. 32 by black boxes. The implementation used a random seed, such that the same test set could be excluded for each execution of the learning procedure.

After the exclusion of the test set, the cross-validation (CV) process can begin. We implemented the K-fold CV process in which the sonar frames were divided into k subsets of size, N/k, where N is the total number of sonar frames. In each iteration of the K-fold CV, $k-1$ subsets were used to generate the ROC plots. The operating point for the ROC plot was selected based on the process described in section 5.3 and the threshold algorithm was evaluated on the k^{th} subset. For example, if $k = 3$, the sonar frames from Fig. 32 are divided into three folds, as shown in Fig. 33. The test set has already been excluded, which is denoted by the black boxes. For each fold, a different subset of frames is left out as the validation subset, which is denoted by the red boxes. The remaining boxes, which retain their same color from Fig. 32, are considered the training frames that are used to create the ROC plots. Each fold results in the creation of an independent ROC plot, as shown in Fig. 34.

After each of the K-folds completes the training and validation phases, the ROC plots

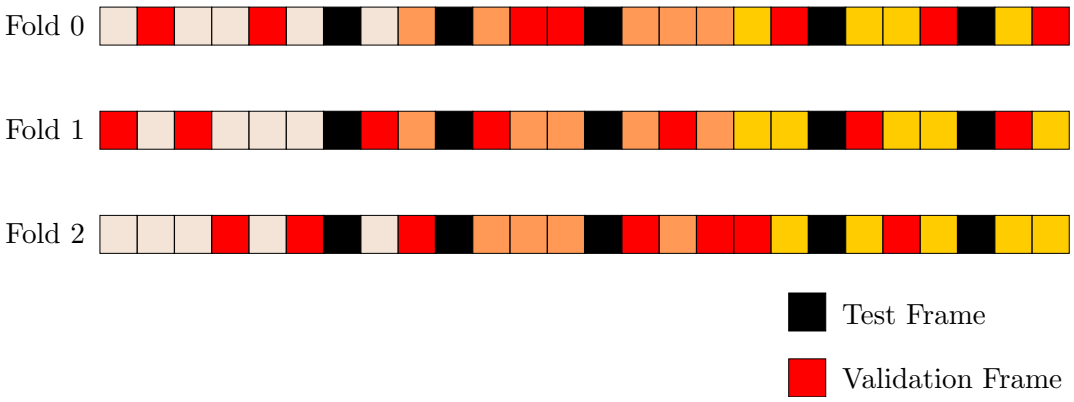


Figure 33: K-folds Cross-Validation

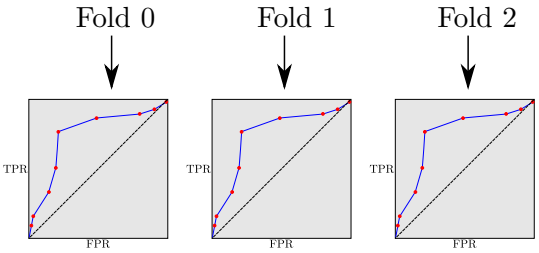


Figure 34: Each fold generates a ROC plot.

from the K folds are "averaged" to create a single ROC plot. The ROC plots are averaged by computing the mean, μ, and standard deviation, σ, of the (FPR, TPR) coordinates at each threshold value.

$$\mu_{FPR} = \frac{1}{T}\sum_{i=1}^{T} FPR_i \qquad \mu_{TPR} = \frac{1}{T}\sum_{i=1}^{T} TPR_i \tag{32}$$

$$\sigma_{FPR} = \sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(FPR_i - \mu_{FPR})^2} \qquad \sigma_{TPR} = \sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(TPR_i - \mu_{TPR})^2} \tag{33}$$

In eq. (32) and eq. (33), T, is the number of discrete threshold values that were used to generate the various operating points. In our framework, each fold utilizes the same threshold values and number of steps. The averaged coordinates, (μ_{FPR}, μ_{TPR}), are then plotted on a ROC plot with their associated 95% confidence intervals, so that the variances between the ROC plots from each fold can be visualized. Finally, the operating point on the averaged ROC plot can be selected based on the procedure described in section 5.3 and used to evaluate the algorithm's performance on the test set. Each of the three threshold algorithms and their selected operating points are evaluated by comparing the statistical

measures of each algorithm (e.g., *accuracy*, TPR, F_1, etc.). While the *accuracy* calculation does not provide an absolute *accuracy* for a classification algorithm on unseen data, by comparing the relative accuracies between the algorithms, we will be able to compare the relative performances among the three threshold algorithms.

5.5 ROC Threshold Algorithm Results

Each of the three threshold algorithms were processed with the K-folds cross-validation framework that was described in section 5.4. During our initial experiments, we found that setting $k = 3$ resulted in decent computational performance and small variance between the ROC plots generated by the folds. The data set was composed of nine different 2D sonar imaging videos. This resulted in a data set of 9,286 sonar frames and 7,188 hand-annotated diver-object bounding boxes. Since 10% of the sonar frames were held-out for the final test set, 929 sonar frames were in the test set and 8,357 sonar frames were used in the K-fold cross-validation subsets.

5.5.1 Static Threshold Results

The static threshold algorithm's ROC plot was generated by sweeping the threshold parameter from 0 to 255 in increments of 10. The averaged ROC plot with two-dimensional 95% confidence intervals for the static threshold is shown in Fig. 35. For the selection of

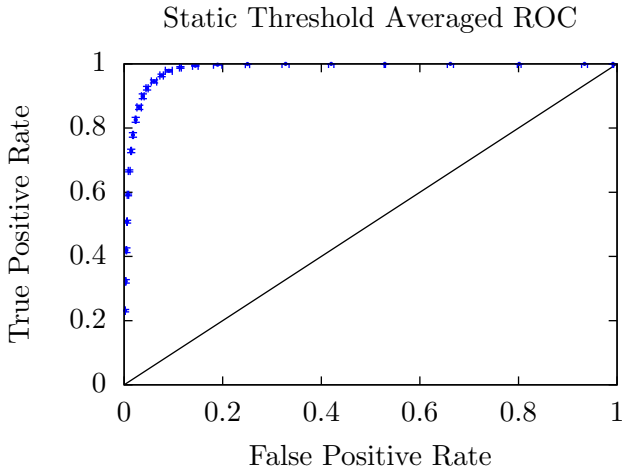


Figure 35: Static Threshold Averaged ROC.

the operating point on the ROC plot, the following costs were specified: $Cost(FP) = 0.5$, $Cost(FN) = 0.5$, $Cost(TP) = 0.0$, and $Cost(TN) = 0.0$. The desired ratio, r_d, of the number of negative to positive samples was specified to be 3. Thus, the slope of the line used to find the operating point was:

$$m = \frac{0.5 - 0.0}{0.5 - 0.0}\frac{3}{1} = 3. \tag{34}$$

This resulted in the iterative line-moving algorithm selecting the point where $FPR = 0.0378367$ and $TPR = 0.899045$, which correlated to a threshold value of 150.

When the static threshold value of 150 was used to evaluate the performance on the test set, the evaluation resulted in an *accuracy* of 94.6188%. The confusion matrix for the static threshold algorithm's classification of objects as being part of a diver or not on the held-out test set is provided in Table 3.

Table 3: Confusion Matrix for Static Threshold on Hold-out Test Set

		Predicted	
		Diver	Not Diver
Actual	Diver	660	65
	Not Diver	91	2083

5.5.2 Gradient Threshold Results

The second threshold algorithm that was evaluated was the gradient-based threshold. The ROC plot was generated by taking the gradient of each sonar frame and then applying the static threshold to the gradient image. Again, the threshold value was swept from 0 to 255 in increments of 10 to create the ROC plot shown in Fig. 36. As with the static threshold operating point selection procedure, the slope of the line was 3. This resulted in the operating point at $FPR = 0.0565366$ and $TPR = 0.879685$ being selected, which correlated to a threshold value of 150.

When the gradient threshold value of 150 was used to evaluate the performance on the test set, the evaluation resulted in an *accuracy* of 93.0939%. The confusion matrix for the gradient threshold algorithm's classification of objects as being part of a diver or not on the held-out test set is provided in Table 4.

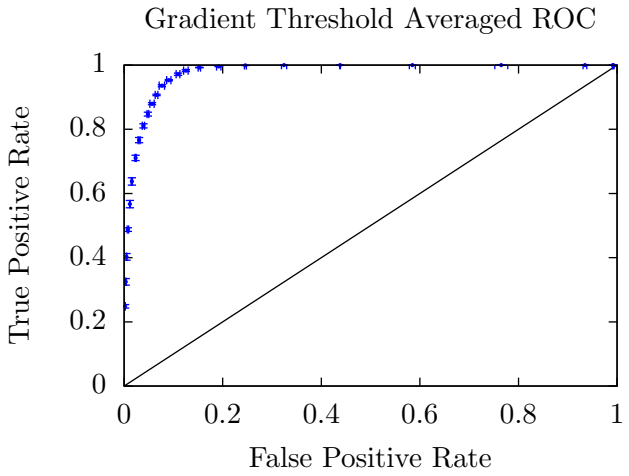


Figure 36: Gradient Threshold Averaged ROC.

Table 4: Confusion Matrix for Gradient Threshold on Hold-out Test Set

		Predicted	
		Diver	Not Diver
Actual	Diver	650	74
	Not Diver	126	2046

5.5.3 Adaptive Threshold Results

Next, the adaptive threshold algorithm was evaluated. The ROC plot was generated by sweeping the desired ratio, Q_d, of the number of non-zero-valued pixels to the total number of pixels from 1E−8 to 9E−3 in increments of 1.8E−4. The relevant ranges for the adaptive threshold algorithm are different from the previous thresholding algorithms because the operating point for the adaptive thresholding algorithm is defined by the ratio of non-zero-valued pixels to zero-valued pixels, while the operating points for the other threshold algorithms were based on a gray scale. The averaged ROC for the adaptive threshold algorithm is shown in Fig. 37. Again, the slope of the line for selecting the operating point was 3, which resulted in selecting the point at $FPR = 0.0407832$ and $TPR = 0.925064$. This operating point correlated with a Q_d of 1.620 01E−3. Since the points in Fig. 37 are tightly spaced, a decimated version of the ROC plot is provided in Fig. 38 to better visualize the 95% confidence intervals of the averaged ROC plot. When the desired ratio of 1.620 01E−3 was used to evaluate the performance on the test set, the evaluation resulted

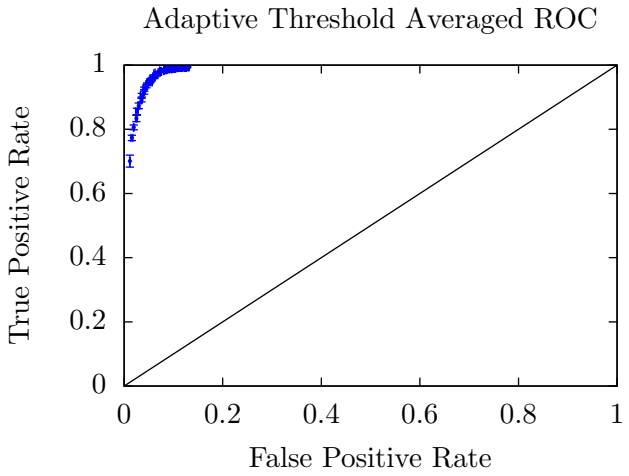


Figure 37: Adaptive Threshold Averaged ROC.

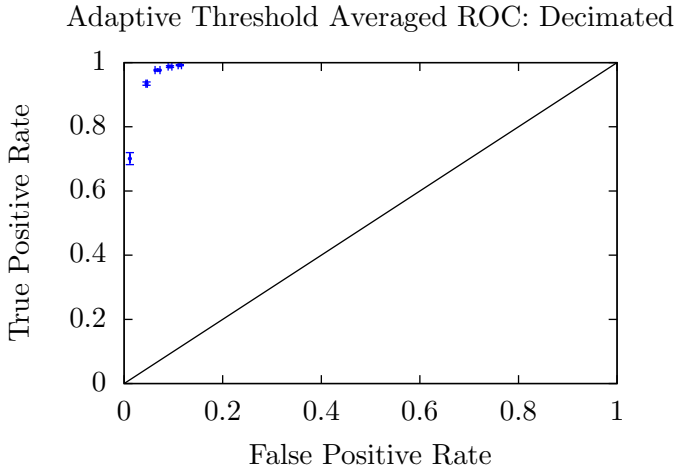


Figure 38: Adaptive Threshold Averaged ROC: Decimated.

in an *accuracy* of 95.9655%. The confusion matrix for the adaptive threshold algorithm's classification of objects as being part of a diver or not on the held-out test set is provided in Table 5.

Table 5: Confusion Matrix for Adaptive Threshold on Hold-out Test Set

		Predicted	
		Diver	Not Diver
Actual	Diver	678	47
	Not Diver	70	2105

5.5.4 Aggregated Results

In addition to FPR, TPR, and *accuracy*, there are a number of other statistical measures that can be used to assess classification algorithms. For a complete comparison of the performances of the three threshold algorithms on the hold-out test set, six different statistical measures were computed and tabulated in Table 6. The PPV is related to the precision of the classification process. The F_1 score can be considered a weighted average of the precision and recall of the classification process.

Table 6: Statistical Measures of Threshold Algorithms on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	$\mathbf{F_1}$
Static	94.62%	91.03%	4.19%	95.81%	87.88%	89.43%
Gradient	93.09%	89.78%	5.80%	94.20%	83.76%	86.66%
Adaptive	95.97%	93.52%	3.22%	96.78%	90.64%	92.06%

5.6 ROC Threshold Algorithm Discussion

Each of the three threshold algorithms produced satisfactory ROC plots and acceptable statistical measures on the final test set. The static and gradient threshold ROC plots produced canonical ROC plots that clearly demonstrated the trade-offs between declaring TPs and FPs by varying the algorithms' threshold values. It is also important to note the small variance in the averaged ROC plots for all threshold algorithms. The small variance is denoted by the closeness of the 95% confidence intervals to the centers of the averaged operating points.

The adaptive threshold's ROC plot is more compressed than the other two algorithms' ROC plots. There are possibly two main reasons for the compression. First, the range of acceptable values to sweep the threshold value over for the static and gradient algorithms was obvious: a gray-scale pixel's value is an integer that can vary from 0 to 255. However, the adaptive threshold algorithm made use of Q_d, the number of non-zero-valued pixels to the total number of pixels. Clearly, the ratio lies within the domain, $\{x \in \mathbb{R} \,|\, 0 \leq x \leq 1\}$. However, the domain had to be limited since iterating over the entire domain was unfeasible. This resulted in an artificially compact grouping of points around the chosen operating point. A second reason why the adaptive threshold algorithm produced a compact grouping of

points could have been that the algorithm is less sensitive to changes in desired ratio values around its "best" operating point.

By comparing the statistical measures in Table 6, it is clear that the adaptive threshold algorithm outperformed both the static and the gradient threshold algorithms in each of the tabulated statistical measures. However, the static threshold did outperform the gradient-based threshold. One issue with the gradient-based threshold is that it discards large areas of high-intensity pixels since the gradient image only highlights changes in pixel intensity.

The adaptive threshold algorithm showed superior performance in each of the six statistical measures that were calculated. Both the sonar hardware's automatic gain controller (AGC) and changes in the underwater scene affect the sonar images' histograms. With a static or gradient-based algorithm, a classifier's threshold value cannot be robustly trained to account for a dynamic scene. However, by adjusting the threshold value in each frame, the adaptive threshold algorithm was able to overcome the scene's changing histogram to classify pixels as being part of an object-of-interest or not.

Still, the diver detection problem has not been solved in this section yet. This section developed and evaluated an adaptive threshold algorithm that classifies pixels as being part of a hand-annotated object-of-interest or part of background "clutter." If the acoustic return of a fish or underwater structure is similar to a diver's acoustic return, the classifier developed in this section could not determine if the pixel belonged to a diver, fish, or underwater structure. However, this section provided us with a well-defined threshold value for the adaptive threshold algorithm.

5.7 ROC Threshold Algorithm Conclusion

In this chapter, we developed a K-fold cross-validation framework for generating ROC plots for three different threshold algorithms: static threshold, gradient threshold, and a novel adaptive threshold algorithm. The ROC plots that were generated from the K-fold cross-validation framework were used to select the operating points for each of the threshold algorithms. The three algorithms where then evaluated on a hold-out test set by comparing the resulting statistical measures (e.g., *accuracy*, TPR, PPV, F_1, etc.). In conclusion, the

adaptive threshold algorithm outperformed the other two threshold algorithms and it will be the primary threshold algorithm used in the sonar image processing pipeline in subsequent chapters. Also, the K-fold cross-validation framework provided us with a well-defined value for the adaptive threshold algorithm's value of Q_d, the desired ratio of the number of non-zero-valued pixels to the number of total pixels in the sonar frame. The adaptive threshold algorithm is the most capable threshold algorithm evaluated in this chapter at adjusting to a dynamic underwater scene and a changing sonar image histogram.

CHAPTER VI

TRACKING OBJECTS IN 2D SONAR IMAGES

6.1 Introduction

In Chapter 5, we developed a K-fold cross-validation framework for comparing three different 2D imaging sonar threshold algorithms. While the analysis of the algorithms is important to the entire diver tracking pipeline, the output of the threshold algorithms is uncorrelated pixels-of-interest. The purpose of this chapter is to describe how we correlate these pixels-of-interest with actual objects in the environment across multiple frames. A flowchart of the sonar image processing pipeline is provided in Fig. 39. The details for implementing

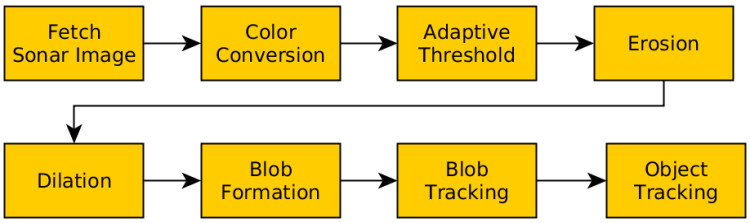


Figure 39: Sonar image processing pipeline.

the first three boxes were provided in Chapter 5. The output of the adaptive threshold algorithm is a collection of uncorrelated pixels. The output of the adaptive threshold algorithm is filtered with erosion and dilation filters to eliminate spurious pixels and noise. After the dilation filter, the remaining pixels are grouped into blobs based on a connected component analysis. Next, the Munkres assignment algorithm associates the blobs from frame-to-frame. The individual blobs are tracked with Kalman filters and are eventually aggregated to build higher-level object trackers. We developed a novel algorithm that adaptively modifies the Kalman filter's measurement matrix, $\boldsymbol{R}$, to better associate new measurements with previous tracks. Finally, we use the age of the lower-level blob tracks to provide a weighted update of the higher-level object's Kalman filter.

6.2 Blob Formation

Blob formation is the process by which uncorrelated pixels are associated into sets of pixels with specific identification values. Before the pixels are associated into blobs, morphological erosion and dilation are applied to a binary image to reduce spurious noise, while maintaining the original structure of the original image [40]. The erosion filter is implemented by sliding a mask, M, of size n by n, across an image. In this case, the shape of M is a circle with radius three pixels, where the pixels in M that are part of the mask are defined as 1 and 0, otherwise. A small circle was chosen, so that the erosion process only removed specular noise without greatly affecting the shape of actual objects in the environment. Also, since most of the objects in the sonar image possessed qualitatively "rounded" features, the circle morphological shape induced the least distortion, compared to a rectangular morphological filter. The elements of the input image region that are selected by M are defined by the set in Eq. 35.

$$A = \{i \in r..(r+n),\ j \in c..(c+n) \mid M(i-r, j-c) > 0\} \tag{35}$$

In Eq. 35, r and c are the current row and column offsets as the mask is slid across the input image. The input pixel that is at the center location of the erosion mask takes on the minimum value, E_A, of the pixels in the set, A.

$$E_A = \min(A) \tag{36}$$

Applying the erosion filter to a binary image has the effect of removing small groups of pixels and thinning lines, spheres, and blobs. For example, the result of applying the erosion filter to the thresholded image in Fig. 40a is shown in Fig. 40b. In Fig. 40b, the erosion filter

(a) Threshold image. (b) Eroded image. (c) Dilated image.

Figure 40: Resulting images from erosion and dilation filter.

removed a thin line of pixels located near the bottom-center of the image. The larger blobs

of pixels passed through the erosion filter without being removed, but they were decreased in size. To return the thinned blobs back to their original sizes, the inverse of the erosion filter, the dilation filter is applied. Similar to the erosion filter, the dilation filter uses the same mask, M, and selects pixel sets as defined in Eq. 35. However, the dilation filter assigns the input pixel that is at the center of the mask to the maximum value, D_A, of the pixel set, A, as denoted in Eq. 37.

$$D_A = \max(A) \tag{37}$$

The result of applying the dilation filter is shown in Fig. 40c. In Fig. 40c, the blobs that were thinned by the erosion filter have almost been returned to their original sizes. Also, the smaller blobs from Fig. 40a have been completely removed from the resulting image. The later stages of blob detection and tracking will be made more computationally feasible by this removal of small pixel groups.

6.3 Blob Data Association and Tracking

The erosion and dilation filters effectively remove noise and small groups of pixels from the sonar image. The purpose of the blob formation step is to associate clusters of neighboring pixels into discrete blobs. A two-pass blob detector, like the one discussed in [64] was developed that associates connected pixels with a number. Our two-pass blob detector requires a binary image. Thus, the output of the dilation process is passed through a static thresholding algorithm to generate a binary image, where the threshold level is set to a value of only one. In the first pass, a kernel, shown in Figure 41, is scanned across the image from top-to-bottom and left-to-right. The values of the pixels in positions "0," "1," "2," and "3" are compared to the value of the pixel in position "4," the current pixel being examined. If the pixels are the same value, they are labeled with the same identifying number. A second pass of the algorithm is used to combine regions that are connected, but had different identifying numbers due to the blob's geometry. During the two-pass blob detector process, the centroid and size of the blobs are calculated for later use in the blob data association process. The centroid, C, of the blob is weighted by the values of the

0	1	2
3	4	

Figure 41: Two Pass Blob Detector Kernel

individual pixels in the blob.

$$C = \frac{\sum p_i I(p_i)}{\sum I(p_i)} \tag{38}$$

The centroid calculation in Eq. 38 requires the coordinate of each pixel, p_i, and the value of each pixel in the image, $I(p_i)$. By weighting the centroid based on the pixel values, we identify the location of the blob that has the highest acoustic reflectivity. It is assumed that this area of highest acoustic return will be present across multiple sonar frames and is the preferred feature to detect. In parallel with calculating each blobs' centroids, the blobs' bounding boxes are also calculated. The bounding box is computed by searching for the minimum and maximum row and column locations of the pixels in each blob. The top-left corner of the bounding box is defined by the blob's minimum row and column positions. The bottom-right corner of the bounding box is defined by the blob's maximum row and column positions.

6.4 Object Data Association and Tracking

The blob formation process resulted in labeled sets of correlated pixels in the sonar image. This is a necessary step for scene understanding. However, the blob formation algorithm only provides unique blob labels for a single frame. A data association algorithm is required to track the blobs across multiple frames. We investigated the use of Munkres' Assignment Algorithm to associate blobs across multiple frames [39]. While originally formulated to optimally assign n persons to n jobs, with respect to the person's ability to perform the job, Munkres' Assignment Algorithm has been used to perform data association for computer vision tasks [59, 10]. Munkres' Assignment Algorithm requires an input cost matrix, $\boldsymbol{C}$,

that specifies the cost for assigning each person to a job.

$$\boldsymbol{C} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix} \tag{39}$$

In eq. 39, $c_{i,j}$ represents the cost of assigning the ith element to the jth element. In our case, instead of calculating the cost of assigning each person to a job, we calculate the cost of assigning the current frame's blobs (i.e., the new "measurements") to the previously instantiated tracks. The Euclidean distances between the centroids of the new measurements and the previous tracks are used to populate the elements of the cost matrix, $\boldsymbol{C}$. If the number of measurements does not match the number of tracks, $\boldsymbol{C}$ is padded with the value of the largest Euclidean distance until $\boldsymbol{C}$ is square. An assignment for the given problem is defined by a set of n independent elements of $\boldsymbol{C}$ (i.e., none of the selected elements are in the same row or column). The optimal assignment, with respect to the Euclidean cost function, is one in which the set of n selected elements results in the smallest total cost, where none of the selected elements lie in the same row or column. Munkres' Algorithm achieves this assignment through a series of row and column reductions until a minimal set of zero-element covering lines is achieved [39]. However, Munkres' Algorithm does not directly account for the creation of new tracks and it can result in assignments that, while optimal with respect to the minimum Euclidean distance, switch track IDs. To account for new track declarations, a concept of a new measurement being "too far" from the previous tracks to be associated with them is required. The validation gate from a Kalman filter is a natural choice for determining whether a new measurement is "too far" from previous tracks.

6.5 Track Filter

Each blob in the sonar image is tracked with a individual Kalman filter [33]. The purpose of the Kalman filter is to use the dynamics of each blob or track to predicate the location of the track in subsequent frames. This is useful when the object being tracked is occluded

for several frames due to other objects or because the object being tracked is entering and exiting the sonar's field-of-view. Objects in the sonar image move slow enough that they can be tracked in the row/column image space and any distortions due to the sonar's range/bearing measurements do not greatly affect the quality of the tracks.

Each target's state, x, is composed of two-dimensional position and velocity information. A constant velocity model is used for each target [47]. The state evolves with time according to the state transition matrix, $\mathbf{\Phi}$, the disturbance matrix, $\mathbf{\Gamma}$, and process white noise, w, as provided in eq. 47.

$$x(k+1) = \mathbf{\Phi}x(k) + \mathbf{\Gamma}w(k) \tag{40}$$

The measurements, z, are related to the state through the measurement matrix, $\boldsymbol{H}$, and measurement white noise, v, as provided in eq. 48.

$$z(k) = \boldsymbol{H}x(k) + v(k) \tag{41}$$

As each sonar frame is processed, the state of each blob is updated based on the Kalman filter time-update equations:

$$\bar{x}(k+1) = \mathbf{\Phi}\hat{x}(k) \tag{42}$$

$$\bar{\boldsymbol{P}}(k+1) = \mathbf{\Phi}\hat{\boldsymbol{P}}(k)\mathbf{\Phi}^T + \mathbf{\Gamma}\boldsymbol{Q}\mathbf{\Gamma}^T \tag{43}$$

$\boldsymbol{Q}$ is the process noise matrix and $\boldsymbol{P}$ is the state covariance matrix. If a newly formed blob is associated with a track, the track's state and covariance are updated based on the measurement-update equations:

$$\boldsymbol{K} = \bar{\boldsymbol{P}}\boldsymbol{H}^T(\boldsymbol{H}\bar{\boldsymbol{P}}\boldsymbol{H}^T + \boldsymbol{R})^{-1} \tag{44}$$

$$\hat{x}(k) = \bar{x}(k) + \boldsymbol{K}(z(k) - \boldsymbol{H}\bar{x}(k)) \tag{45}$$

$$\hat{\boldsymbol{P}}(k) = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\bar{\boldsymbol{P}}, \tag{46}$$

where $\boldsymbol{R}$ is the measurement noise matrix, $\boldsymbol{K}$ is the Kalman gain matrix, and $\boldsymbol{I}$ is the identity matrix. Since we assumed constant velocity models for the blobs in our sonar

images the state transition matrix is defined as

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{47}$$

Also, the measurements come in the form of row/column position, so the measurement matrix is

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{48}$$

Finally, the process noise matrix, measurement noise matrix, and disturbance matrix are

$$\boldsymbol{Q} = \begin{bmatrix} qT & 0 \\ 0 & qT \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \boldsymbol{\Gamma} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{49}$$

The 2D imaging sonar used during data collection had a frame rate of 15 Hz. This results in a time-update of $T = 1/15$ seconds. Based on empirical experimentation, we found that values of $r = 10$ and $q = 1$ result in proper track convergence.

In parallel with updating the Kalman filter equations for each track, two age counters are also maintained to assist in culling dead tracks. Each time a new measurement is associated with a previously detected track, an "alive" counter is incremented for that track. If a previously detected track does not have a measurement associated with it in a specific frame, the track's "occluded" counter is incremented. If the occluded counter reaches a certain, the track is deemed "dead" and is culled from the valid track list. The occluded counter is a parameter that has to be tuned for a given application, similar to the Kalman filter's $\boldsymbol{R}$ and $\boldsymbol{Q}$ matrices. For our application, the occluded counter was tuned to a value of 11 frames.

The Kalman filter framework also provides a means for determining if a new measurement is within a certain number of standard deviations from the mean of the track's position.

Given the measurement noise covariance matrix,

$$\boldsymbol{B} = \boldsymbol{H}\bar{\boldsymbol{P}}\boldsymbol{H}^T + \boldsymbol{R}, \tag{50}$$

a measurement, Z_m, falls within n number of standard deviations of the track's position's mean if

$$(Z_m - \boldsymbol{H}\bar{x})^T \boldsymbol{B}^{-1}(Z_m - \boldsymbol{H}\bar{x}) \leq n. \tag{51}$$

We can use the gate validation region provided by eq. 51 as a means to determine if a measurement is "too far" or "close enough" to a previously declared track. This solves the problem that we encountered at the end of section 6.4 due to the switching of track IDs using Munkres' algorithm. In addition, if the new measurement is $n = 3$ standard deviations outside of the validation region of the track that Munkres' Algorithm selected, a new track is declared.

6.6 Blob Track Consolidation

The Munkres' Assignment Algorithm plus the Kalman filter-based approach we developed in the previous sections are able to track blobs that are present from frame-to-frame and even when they are occluded for several frames. However, even static objects can appear fragmented in the sonar images due to the noisy nature of underwater acoustics. To account for this fragmentation at the blob level, we implemented an algorithm that consolidates blob tracks by efficiently searching for overlapping blob bounding boxes. The assumption is that if a newly detected blob is not associated with a previous track, but the new blob's bounding box intersects with the bounding box of a previous track, the new blob should be used as a measurement to update the previous track. The Consolidate Blob Tracks (CBT) algorithm is provided in Algorithm 2. The CBT algorithm takes a *tracks* list as an input. This input list can consist of old and new tracks. The CBT algorithm returns a list of $fused$ tracks, where tracks with overlapping bounding boxes have been consolidated into a single track. Also, tracks that did not overlap are copied over into the returned $fused$ list. As previously discussed, each track has an associated Kalman filter, age, and bounding box. Also, for this algorithm, each track object is augmented with a Boolean value to designate

Algorithm 2 Consolidate Blob Tracks

Precondition: *tracks* is a list with each blob's track information
Postcondition: $fused$ is a list of consolidated blobs from *tracks*

```
1: function ConsolidateBlobTracks(tracks)
2:     id_next ← 0
3:     for each t_1 in tracks do
4:         for each t_2 in tracks do
5:             if t_1 == t_2 then
6:                 no op
7:             else if overlap(t_1,t_2) then
8:                 if is_matched(t_1) and is_matched(t_2) then
9:                     if matched_id(t_1) < matched_id(t_2) then
10:                        append(map(matched_id(t_1)),map(matched_id(t_2)))
11:                        clear(map(matched_id(t_2)))
12:                    else
13:                        append(map(matched_id(t_2)),map(matched_id(t_1)))
14:                        clear(map(matched_id(t_1)))
15:                    end if
16:                else if is_matched(t_1) then
17:                    set_matched_id(t_2, matched_id(t_1))
18:                    append(map(matched_id(t_1)), t_2)
19:                else if is_matched(t_2) then
20:                    set_matched_id(t_1, matched_id(t_2))
21:                    append(map(matched_id(t_2)), t_1)
22:                else
23:                    set_matched_id(t_1, id_next)
24:                    set_matched_id(t_2, id_next)
25:                    append(map(id_next), t_1)
26:                    append(map(id_next), t_2)
27:                    id_next ← id_next + 1
28:                end if
29:            end if
30:        end for
31:    end for
32:
33:    for each list in map do
34:        t_old ← find_oldest_track(list)
35:        update_track(t_old, list)
36:        append(fused, t_old)
37:    end for
38:
39:    for each t_1 in tracks do
40:        if not is_matched(t_1) then
41:            append(fused, t_1)
42:        end if
43:    end for
44:    return fused
45: end function
```

whether the track has been "matched" with another overlapping track. The status of this Boolean value can be queried by calling the function, is_matched(), on the track object. The actual ID of the matched track can be queried by calling matched_id() on the track object. Another important concept for this algorithm is that a hash map is used to store lists of overlapping tracks. For example, the function call, map(3), provides access to a list of tracks associated with ID number 3. Finally, the ID is not the track ID, but a temporary ID used to associate overlapping tracks. Thus, the overall CBT algorithm has three main stages: finding overlapping bounding boxes and associating them with the same ID, finding the oldest track in each ID list, and adding tracks that did not overlap with any other tracks to the output list of $fused$ tracks.

The first stage in the CBT algorithm involves comparing each of the tracks to each other to determine if there is any overlap. Even if the second **for** loop is optimized by only considering elements in the $tracks$ list after t_1, this double loop operation is still an $\mathcal{O}(n^2)$ operation. The overlap() function returns true if the two input tracks have overlapping bounding boxes, otherwise, it returns false. The overlap() function can operate efficiently if each track's bounding box maintains maximum and minimum values for its row and column information. If one of the bounding box's maximum column value is smaller than the other bounding box's minimum column value, then the boxes cannot overlap. Likewise, if one of the bounding box's maximum row values is smaller than the other bounding box's minimum row value, then the boxes cannot overlap. If the two previous statements were not true, then the two bounding boxes must overlap. Once it is determined that an overlap exists, see line 7, the is_matched() function is called to determine if either of the tracks have an associated ID. If both tracks have matched IDs, the list from the larger ID is copied to the list with the smaller ID. This copying takes place through the append() function, where the first argument is the destination list and the second argument is the source list. If only one of the two tracks is already matched, the unmatched track is appended to the list of the already matched track. This takes place in lines 16 through 21. Finally, the last *else* statement at line 22 accounts for the case where there is an overlap, but none of the tracks have been matched yet. In this case, the next association ID, id_{next}, is added to the map

and the two tracks are appended to its list.

After the tracks with overlapping bounding boxes have been associated with IDs, the oldest track in each ID list is found, shown at line 34. The update_track() function then calls the appropriate Kalman filter update function on the oldest track, t_{old}, where the rest of the list provides the Kalman filter measurements. After t_{old} has been updated to account for overlapping tracks, it is appended to the $fused$ output list. Finally, the tracks that did not overlap with any other bounding boxes, (i.e., the is_matched() function should return false when called on them), are appended to the $fused$ output list at line 41.

6.7 Object Tracker

6.7.1 Introduction

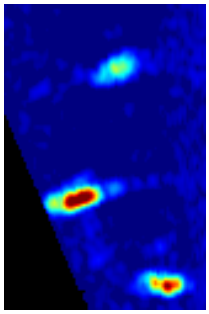
In the previous section, we developed an algorithm for tracking multiple blobs across multiple frames. Tracking the blobs allows us to maintain information about the blobs, such as blob sizes, blob ages, blob trajectories, and blob positions. However, at this point, the blobs are not associated with any specific objects. Also, it is important to note that the 2D sonar image pre-processing steps result in many fragmented and disconnected blobs that are actually from the same physical object. This is a primary difference between traditional multi-hypothesis tracking problems and the one discussed in this work. Scuba divers are especially prone to returning many fragmented returns when ensonified with a 2D imaging sonar. For example, the abdomen, hands, and fins of the diver shown in Fig. 42a, provide a very strong acoustic returns. However, the arms and legs of the diver provide poor returns.

(a) Scuba Diver in 2D imaging sonar frame. (b) Scuba Diver's fragmented blob returns.

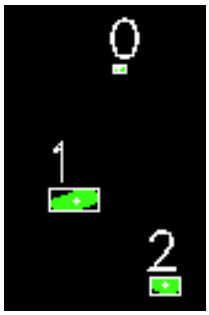
Figure 42: 2D sonar image pre-processing pipeline results in fragmented blob returns.

This results in an image pre-processing pipeline that generates multiple disconnected blobs, shown in Fig. 42. Another complication is that there are some objects in the underwater

scenes that do only generate a single blob. These objects might be small fish, small underwater structures, or coral reefs. During our data collection at NASA NEEMO, we were



(a) Sonar image of three fish recorded at NASA NEEMO.



(b) Extracted acoustic blobs from three fish.

Figure 43: Sonar image pre-processing of smaller objects, such as fish, might produce single blobs for each object.

able to record 2D imaging sonar data of multiple fish. The portion of the original sonar frame that contains the acoustic returns of the three fish is shown in Fig. 43a. Due to the fact a fish's body structure is much simpler, geometrically, from that of a human scuba diver's body structure, a single acoustic blob is returned from each fish after the image pre-processing pipeline operates on the sonar image. An image of the three blobs that are returned for the three fish is shown in Fig. 43b. The fact that the sonar image pre-processing pipeline generates fragmented blob measurements from some objects, but connected blob measurements for others, makes the process of uniquely identifying physical objects in the scene difficult. Thus, we need to develop a multiple object tracker that can handle both cases: associating fragmented blobs returns with certain objects and associating single blob returns for others, where appropriate.

6.7.2 Multiple Object Tracker Algorithm

At the heart of our multiple object tracker (MOT) is the Kalman filter. However, we have developed a novel method for dynamically warping the measurement matrix, $\boldsymbol{R}$, during run-time, to account for objects that produce multiple acoustic returns. The method also handles the case of tracking objects that only produce single blob measurements.

The MOT's overall process involves maintaining a list of object tracks. The input to

the MOT is a list of blob tracks, which are considered measurements as far as the MOT is concerned. The blob tracks have their own associated estimated centroid's, Kalman filter's, and ages that were fused during the earlier blob tracking stages. The six primary steps in the MOT's process involved:

1. Predicting the locations of the objects using the Kalman filter's motion model.
2. Determining if a new blob track measurement falls within the gate of an object tracker.
3. Combining blob track measurements that fall within similar object tracks' gates.
4. Distorting the Kalman filter measurement matrix, $\boldsymbol{R}$, to coincide with the covariance matrix of the blob track measurement's centroid.
5. Combining object tracks that have similar track centroids.
6. Culling dead tracks.

The MOT process is very similar to the lower-level blob tracking process, but with some important distinctions. The first difference is that Munkres' assignment algorithm is not used to associate new measurements with the already existing tracks. Instead, if a measurement falls within the gate of an object's Kalman filter, it is assumed to be associated with that object's tracker. Thus, a single measurement could be associated with two different object trackers if the measurement fell within the gates of both object trackers. While this does not result in perfectly assigned measurement-to-track configurations, it results in configurations that are never completely wrong. This approach is similar to the Joint Probabilistic Data Association Filter (JPDAF) algorithm [6]. The second difference is that the Kalman filter's measurement matrix, $\boldsymbol{R}$, is adaptively modified to account for the fact that parts of the scuba diver's body that provide strong acoustic returns become occluded as the diver articulates their arms and legs. Since the diver's arms and legs often become detached after the image pre-processing stages, with a traditional tracker, the blobs associated with the arms and legs will cause the generation of new object tracks. To account for this, the measurement matrix could be enlarged to capture the new measurements. However, this

can result in an artificially large measurement matrix that consumes measurements that should not be associated with the scuba diver. Instead, we developed a method that dynamically distorts the measurement matrix to fit to an ellipse around the scuba diver's body structure.

The MOT process begins similarly to the blob tracking process: predicting the positions of the list of object tracks based on each track's Kalman filter. The Kalman filter time-update equations for the estimated state and covariance matrices have already been provided in eq. 42 and eq. 43. Next, the list of "alive" blob track measurements is processed. If a blob track is both currently alive and has had a measurement associated with it in the current frame, it is determined whether it's centroid falls within the gates of any of the previously existing object tracks. Again, the gate, or validation region, was previously provided in eq. 51. If the measurement does not fall within the gate of any previously existing tracks, a new track is instantiated where the measurement's position is used to initialize the new track.

After associating the new measurements to the existing tracks, there are two cases to consider:

1. The object tracker was associated with one or more measurements.

2. The object tracker was not associated with any new measurements.

Updating the existing track is simple if only a single measurement is associated with a track. The measurement's centroid is used in the Kalman filter's measurement-update equations. However, if multiple measurements fall within the gate of a single object track, a question arises, "how do we combine the measurements to provide a measurement-update for the object's Kalman filter?" We could use all the measurements as inputs to the standard Kalman filter measurement-update step. However, the associated age for each blob track provides a secondary track confidence and we want to take age into account when updating the object tracker. We find the age-weighted centroid of the combined measurements, C_{m_w}, by weighting the centroid by the ages of the blobs' tracks, as shown in eq. 52.

$$C_{m_w} = \frac{\sum_{i=1}^{M} B_c^i B_a^i}{\sum_{i=1}^{M} B_a^i} \tag{52}$$

M is the number of measurements that fall within a specific object tracker's gate, B_c^i is the i^{th} blob measurement's centroid, and B_a^i is the i^{th} blob measurement's age.

In addition to calculating the centroid of the measurements, the MOT process deviates from the traditional tracking process by distorting the tracker's measurement matrix, $\boldsymbol{R}$, to account for the distribution of the blob tracks' positions that were used to create C_{m_w}. In parallel with calculating the age-weighted centroid, C_{m_w}, the unweighted measurement centroid, C_m is also calculated, as shown in eq. 53.

$$C_m = \frac{1}{M}\sum_{i=1}^{M} B_c^i \tag{53}$$

Next, the measurement position covariance matrix, Σ_m, is calculated based on eq. 54.

$$\boldsymbol{\Sigma}_m = \frac{1}{M}\sum_{i=1}^{M}(B_c^i - C_m)(B_c^i - C_m)^T \tag{54}$$

However, $\boldsymbol{\Sigma}_m$ can become singular if there is a small number of blobs or if they lie on a line. Thus, we use eigendecomposition to extract the eigenvalues from $\boldsymbol{\Sigma}_m$ and modify the eigenvalues before reconstructing the matrix. The process involves factorizing $\boldsymbol{\Sigma}_m$ such that

$$\boldsymbol{\Sigma}_m = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{55}$$

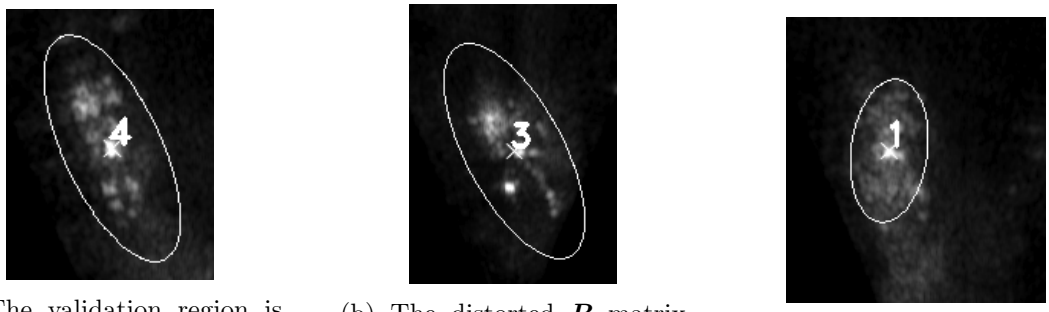
where $\boldsymbol{Q}$ is a square matrix containing the eigenvectors of $\boldsymbol{\Sigma}_m$ and $\boldsymbol{\Lambda}$ is a square matrix with $\boldsymbol{\Sigma}_m$'s eigenvalues along its diagonal. If the smaller of the two eigenvalues is less than 0.1 times the larger eigenvalue, the smaller eigenvalue is set to 0.1 times the larger eigenvalue. $\boldsymbol{\Sigma}_m$ can then be reconstructed by populating $\boldsymbol{\Lambda}$ with the new eigenvalues and using eq. 55. This process ensures that $\boldsymbol{\Sigma}_m$ does not approach a singularity [11]. For the case that only a single blob measurement was associated with the object tracker, $\boldsymbol{\Sigma}_m$ cannot be constructed. Thus, to encourage $\boldsymbol{R}$ to not be any larger than is required to encompass the object's blob measurements, the eigenvalues of $\boldsymbol{R}$ are multiplied by 0.90 and used to construct a pseudo $\boldsymbol{\Sigma}_m$.

At this point, $\boldsymbol{\Sigma}_m$ encompasses the spacial distribution of the blob measurements that were used to update the object tracker. However, $\boldsymbol{\Sigma}_m$ is not used directly to update the object's measurement matrix, $\boldsymbol{R}$. This could result in vastly different validation regions from frame-to-frame due to the random occlusions of the diver's fins, hands, and head.

Instead, $\boldsymbol{\Sigma}_m$ is used as a measurement to update a Kalman filter that provides an estimate of the object track's $\boldsymbol{R}$ matrix. The $\boldsymbol{R}$ matrix tracker consists of four independent Kalman filters that track scalar values. Thus, the associated Kalman filter matrices for each scalar tracker are simply

$$\boldsymbol{\Phi} = 1 \quad \boldsymbol{H} = 1 \quad \boldsymbol{Q} = 0.01 \quad \boldsymbol{R} = 2 \quad \boldsymbol{\Gamma} = 0. \tag{56}$$

The result of applying this technique of using the object's blob measurements to adaptively update the object's $\boldsymbol{R}$ matrix is shown in Fig. 44. As blob measurements are associated with the object tracker, the distorted $\boldsymbol{R}$ matrix fits the validation region to the diver's outline, as shown in Fig. 44a. A difficult tracking example is shown in Fig. 44c due to the low-contrast in the sonar image. However, the object tracker that we developed was able to fit its measurement validation region to the diver's general outline.



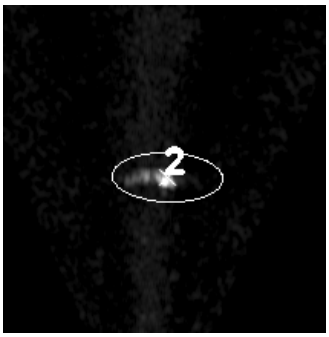
(a) The validation region is elongated along the length of the diver's body.

(b) The distorted $\boldsymbol{R}$ matrix fits the validation region to the diver's outline.

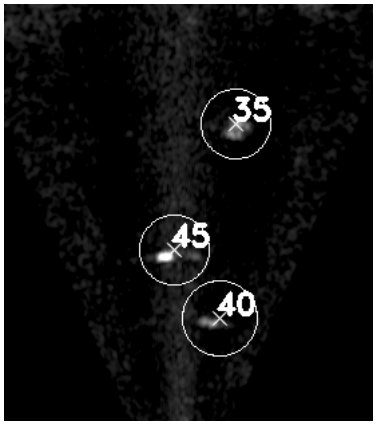
(c) Tracking the diver object in a low-contrast scenario.

Figure 44: The adaptive $\boldsymbol{R}$ matrix algorithm distorts the measurement validation region around the diver to encompass the diver's arms and legs.

Even though the primary objective of this work is to detect and track divers, it is proposed that by tracking all objects in the scene, diver or not, the diver track quality will increase (i.e., the diver tracks will experience fewer track ID changes because non-diver clutter will be tracked with an ID). Thus, to demonstrate the effectiveness of our multi-object tracker we tested the tracker on 2D imaging sonar data of fish that we collected at the NASA NEEMO 20 mission. Unlike divers, fish do not generate fragmented blob measurements when ensonified since they are more compact than divers. An example of a single fish being tracked is shown in Fig. 45a. Our multi-object tracker correctly associates blob measurements across multiple small objects that do not produce fragmented blobs as



(a) The tracker's measurement validation region is elongated along the major axis of the fish.



(b) The distorted $\boldsymbol{R}$ matrix fits the validation region to the diver's outline.

Figure 45: The adaptive $\boldsymbol{R}$ matrix algorithm fits the measurement validation region to non-fragmenting objects as well.

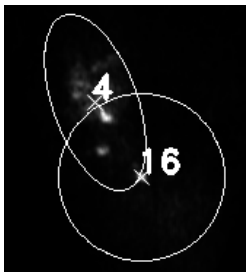
well, as shown in Fig. 45b.

Despite the distortions that the adaptive $\boldsymbol{R}$ matrix algorithm applies to the measurement validation region, some blob measurements from a fragmenting object fall outside of the measurement validation region. This results in the generation of a separate and erroneous object track. However, by updating future blob measurements that fall within the erroneous track's gate in parallel with the measurements that fall within the actual track's gate, the two object tracks tend to converge toward the same centroid. Thus, when multiple object tracks' three standard deviation validation regions contain each other's estimated centroids, the object tracks are merged. The age of the merged track is that of the oldest merged track and the other younger tracks' centroids are used to update the Kalman filter of the older track. An example of this merging process taking place is shown in Fig. 46. This merging process reduces the number of track ID changes during a scenario by favoring tracks with earlier births.

6.8 Results

We conducted three different experiments in which 2D imaging sonar data files were processed with our multiple fragmented object tracker.

(a) A single object track exists.



(b) An erroneous object track is instantiated just outside the valid track's gate.

(c) The validation region of the erroneous track becomes similar to the valid track's gate.

(d) The erroneous track moves towards the centroid of the valid track.

(e) The erroneous track becomes more similar to the valid track.

(f) The erroneous track is consumed by the valid track and the original track ID has been preserved.

Figure 46: An erroneous track will be merged back into the original valid track if the two object tracks are similar.

6.8.1 Experiment #1

In the first experiment, a swimming scuba diver was ensonified with a stationary 2D imaging sonar in the Georgia Tech acoustic dive well. Plots of the tracked blobs trajectories for experiment #1 are shown in Fig. 47. Plots of the tracked object trajectories for experiment #1 are shown in Fig. 48. While the blobs and objects are tracked in two-dimensions, the trajectories are plotted in three dimensions to emphasize the change in position over time. The third dimension is the frame number of the sonar data file. Also, each color and plot point type denotes a different blob or object ID number.

6.8.2 Experiment #2

Data for the second experiment was collected at NASA NEEMO Mission 20 in which two aquanauts were ensonified with a 2D imaging sonar attached to the Naval Postgraduate School's Agile Close-Quarters Underwater Autonomous System (ACQUAS) [21]. The two

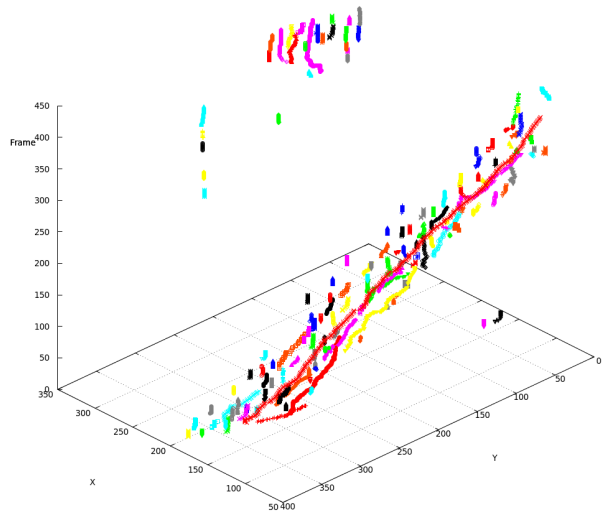


Figure 47: Experiment #1: Blob tracks for swimming scuba diver.

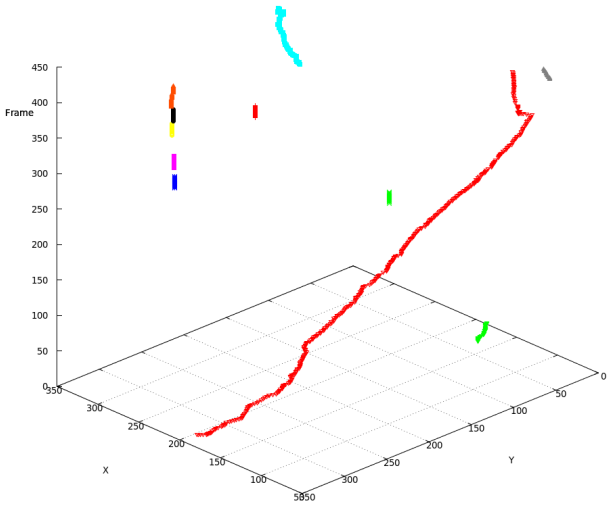


Figure 48: Experiment #1: Object tracks for swimming scuba diver.

aquanauts were wearing environmental suits and walking in the vicinity of a man-made structure. Plots of the tracked blobs trajectories for experiment #2 are shown in Fig. 49. Plots of the tracked object trajectories for experiment #2 are shown in Fig. 50.

6.8.3 Experiment #3

The third experiment consisted of a swimming scuba diver in the Georgia Tech acoustic dive well, but in the presence of a man-made structure at the bottom of the dive well. Also, in the third experiment, the sonar head was rotated during data collection to test the tracker's

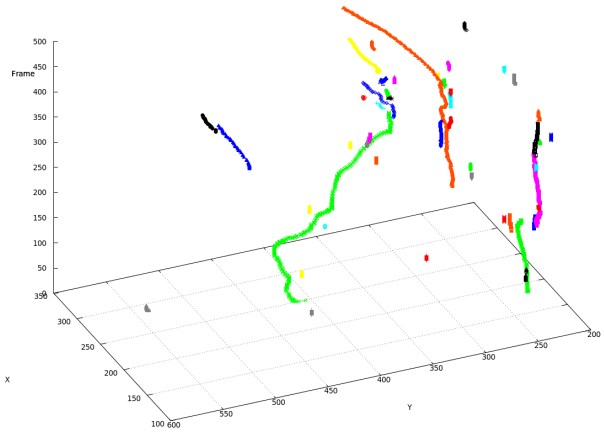


Figure 49: Experiment #2: Blob tracks for two walking aquanauts at NASA NEEMO mission 20.

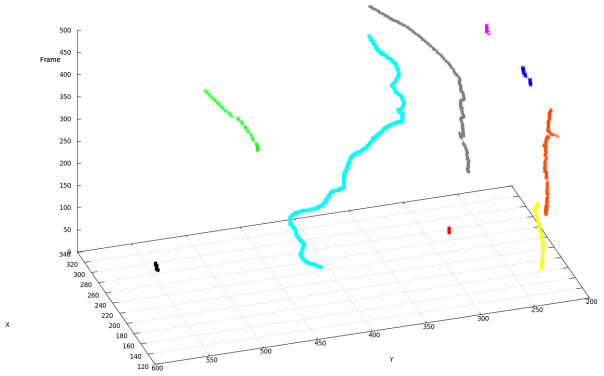


Figure 50: Experiment #2: Object tracks for two walking aquanauts at NASA NEEMO mission 20.

robustness to nonlinear motion. Plots of the tracked blobs trajectories for experiment #3 are shown in Fig. 51. Plots of the tracked object trajectories for experiment #3 are shown in Fig. 52.

6.9 Discussion

Two plots were provided for each experiment. The importance of the tracked blob trajectories plots was to demonstrate that ensonifying divers with 2D imaging sonar generated multiple fragmented returns. This effect is obvious in Fig. 47, where a long cluster of blob tracks progresses from the bottom-middle to the top-right of the plot. This long cluster of blob tracks is the swimming scuba diver. Also, it can be noted that even with Munkres' Assignment Algorithm and individual Kalman filters for each blob, the blobs are difficult to

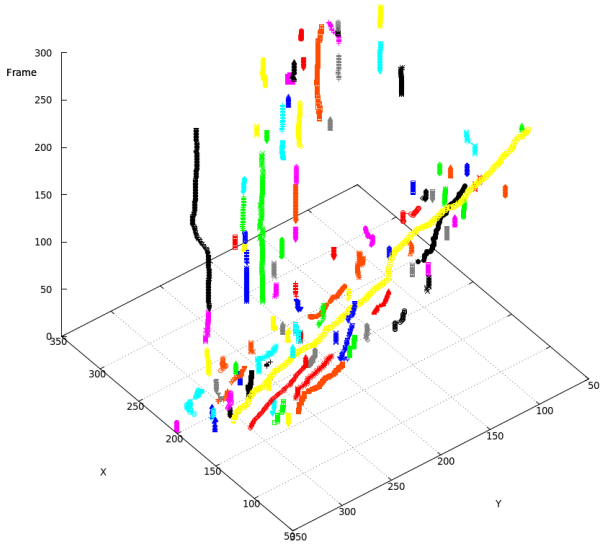


Figure 51: Experiment #3: Blob tracks for swimming scuba diver and rotating sonar head.

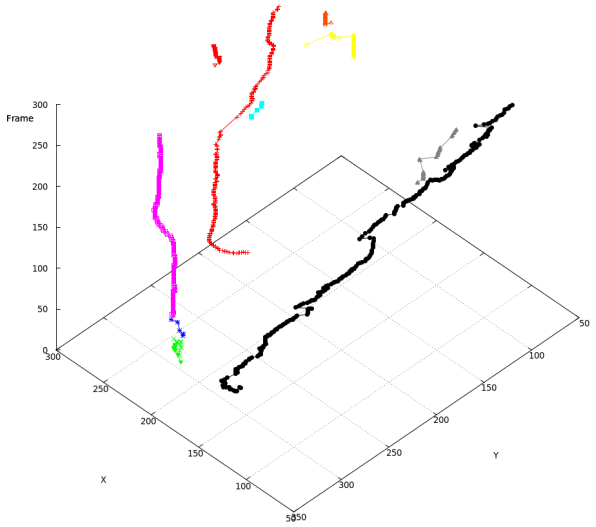


Figure 52: Experiment #3: Object tracks for swimming scuba diver and rotating sonar head.

track for long periods of time. This is due to the fact that fragmented blob returns around the diver move in and out of the sonar's cone of detection. Much cleaner and longer-term trajectories can be found in the object trajectories in Fig. 48. The success of these cleaner trajectories is due to the adaptive Kalman filter $\boldsymbol{R}$ measurement matrix algorithm.

The effectiveness of the multiple fragmented object tracker to track two aquanauts, a man-made structure, and a fish is shown in Fig. 49 and Fig. 50. The blob tracker provides longer continuous tracks when tracking the walking aquanauts in Fig. 49 than when tracking

swimming scuba divers, as in Fig. 47. An explanation for this is that, even though a 2D imaging sonar has a three-dimensional detection cone, 2D imaging sonar is very similar to a scanning laser range finder in that it has a plane of maximum detection. The orientation of the detection plane and the object being detected determines the types of returns that are generated. Since a swimming scuba diver is horizontally-oriented in the water column, 2D imaging sonar will be able to detect the diver's fins, abdomen, and head, separately. However, for the case of the vertically-oriented walking aquanauts, 2D imaging sonar can only detect a single planar slice of the aquanauts abdomen, legs, or head. Also, a swimming scuba diver is less encumbered with equipment; thus, they move their arms and legs more quickly than an aquanaut. This can result in the diver's arms and legs moving in and out of the sonar's detection cone more frequently, causing the generation and destruction of short-lived blob tracks. In Fig. 50, the object tracks of a fish, the small green trajectory located in the left of the plot; a diver, the light-green trajectory located in the middle of the plot; and a second diver, the grey trajectory located in the middle-right of the plot are shown. The other object trajectories in Fig. 50 were generated by the detection of plant-life and mooring lines around the NASA NEEMO Aquarius habitat.

As in experiment #1, the multiple object tracker aggregates the tracked blobs, shown in Fig. 51, into tracked objects with longer continuous trajectories, shown in Fig. 52, in experiment #3. In experiment #3, the sonar head was rotated during data collection. The point in time at which the sonar head was rotated can be visualized by looking at the middle of the purple trajectory on the left side of the plot in Fig. 52. Even with this rotation, the multiple object tracker was able to provide continuous tracks of the purple, red, and black trajectories. In this experiment, the black trajectory was the diver, the purple trajectory was a man-made structure at the bottom of the dive well, and the red trajectory was a structure on the side of the dive well.

6.10 Conclusion

In this chapter we have developed a multiple object tracker that is capable of tracking objects that produce fragmented returns when ensonified. The fact that the types of objects that we

are tracking (e.g., divers) produce multiple returns required us to develop an augmentation to the classical multi-hypothesis tracking approach. To account for the fragmented object returns, a novel adaptive Kalman filter $\boldsymbol{R}$ measurement matrix algorithm was developed. By distorting the $\boldsymbol{R}$ matrix and, thus, the measurement validation region, the object tracker was able to "catch" the fragmented blobs that belonged to it. Also, we demonstrated that the same object tracking algorithm can be used to track objects that do not produce fragmented returns, such as fish.

Just to be able to provide consistent track IDs for generic objects in 2D imaging sonar data, we have had to develop layers of tracking filters. However, the multiple object tracker does not provide any direct means of classifying the objects that are being tracked. In the next chapter, we develop and assess features in the 2D imaging sonar data that can be used to classify the object tracks.

CHAPTER VII

DIVER CLASSIFICATION IN 2D IMAGING SONAR

7.1 Introduction

The primary purpose of this research is to develop algorithms to detect and track a human diver in 2D imaging sonar. In Chapter 6, we developed the algorithms that group pixels into meaningful objects that can be tracked across multiple frames. In this chapter, we develop the algorithms that classify the generic objects as either being a human diver or not a human diver. However, this classification task is made difficult by the fact that when a human diver is ensonified with a 2D imaging sonar, it is difficult for even a trained imaging sonar operator to distinguish divers from static objects in a single frame. Instead, a human operator will make use of the temporal changes across multiple frames to classify an object in imaging sonar.

A first-order feature for classifying an object can be obtained by detecting whether an object has an estimated velocity above a given threshold. This method is clearly susceptible to false positives from moving objects in the scene that are not divers, but it will be used as a baseline to compare against our novel diver feature detector that is based on tracking the diver's two fins with respect to the diver's body. We use Receiver Operating Characteristic (ROC) analysis to compare the results of the velocity-based and fin tracking classifiers. The same hand-annotated data set is used to stimulate both classifiers and includes 3,231 frames of positive diver objects. There are an additional 607 sonar frames of negative samples that do not contain any divers, but do include moving objects, such as fish. In addition to using ROC analysis to compare classifiers, we also use the ROC analysis to tune the various parameters required by the classifiers. Care was taken to ensure that the data set was properly, and randomly, divided into training, cross-validation, and testing data sets. The same K-folds framework that was developed in Chapter 5 was used in the development and testing of the classifiers.

7.2 Classification via Tracking of Diver Fins

In computer vision classification applications, it is typical to build an object classifier by extracting features from many positive samples of the object-of-interest and then "learn" the relationships between the features that define the object. This is the approach taken by the Support Vector Machines / Histogram of Oriented Gradients (HOG+SVM) method [8]. However, the lack of large data sets and the fact that ensonifying a diver with 2D imaging sonar produces amorphous shapes, means that it would be very difficult to build a HOG+SVM classifier for human divers in 2D imaging sonar. Instead, we decided to classify objects in the sonar data based on temporal changes and by assuming that ensonifying a diver produces multiple fragmented returns. Specifically, when ensonified, a diver object produces a large return at the centroid of the diver and two returns behind the diver, which represent the diver's fins. As the diver kicks their fins to produce movement, the fins move in and out of the sonar's cone-of-detection. The fin tracking classifier attempts to track these temporal fin kicks, using the covariance matrices in the fin trackers as measures of classification confidence. We use an assumed model of the diver's movement and structure to build a diver classifier.

7.2.1 Method

The diver fin tracker classifier operates by attempting to track two objects behind the object's direction of motion, but within the object's Kalman filter error ellipse. As the diver's fins move in and out of the sonar's detection cone, new measurements will update the fin trackers. If the object is in fact a diver, the covariance matrices of the two fin trackers will decrease in magnitude and eventually shrink below a specific covariance matrix threshold, P_{th}. This method does require the object to have a minimum forward velocity, v_{min}, since the classifier has to search for fins in the opposite direction of the object's forward velocity. In order for a new blob measurement to be associated with one of the fin trackers, the following conditions must be met:

1. The object must be moving with a minimum velocity, v_{min}.

2. The blob measurement possibly representing the fin must be visible in the current frame. (i.e., the predicted locations of occluded blobs are not used to update fin trackers.)

3. The possible fin measurement must be located within the error ellipse of the object to be classified. Eq. 51 is used to determine whether the measurement falls within 3 standard deviations of the object's centroid.

4. The norm of the vector from the object's centroid to the measurement has to be greater than a minimum leg-length threshold, ll_{th}. This is to ensure that the object's centroid is not used to update the fin trackers.

5. The measurement must be located in the "back half" of the object's error ellipse with respect to the object's velocity. The measurement is in the "back half" of the object if the dot product of the object's velocity, $\vec{v_f}$, and the vector pointing from the object's centroid to the measurement, denoted by $\vec{m}$, is less than zero, (i.e., the measurement is in the back half if $\vec{v_f} \bullet \vec{m} < 0$). An example geometric configuration of a measurement that falls within the back half of an object's error ellipse is shown in Fig. 53.

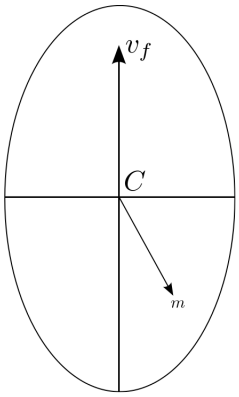


Figure 53: An example of a measurement falling in the "back half" of the object's error ellipse.

If the previous requirements are satisfied for a given measurement, then the measurement is added to a list of measurements that will possibly be used to update either the left or the right fin trackers. The locations of the predicted fin trackers relative to the diver's centroid,

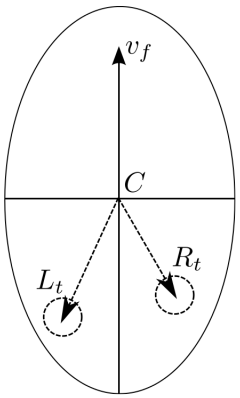


Figure 54: Locations of the left and right fin trackers relative to the object's centroid and velocity.

C, are shown in Fig. 54. In Fig. 54, note that the left and right trackers, L_t and R_t, are on the opposite end of the object track's error ellipse with respect to the object's forward velocity, $\vec{v_f}$. To determine whether a measurement is added to the list of measurements associated with the left or right tracker, the cross product of the negative forward velocity, $-\vec{v_f}$, and the vector pointing from the object's centroid to the measurement's position, $\vec{m}$, is evaluated. Since $\vec{v_f}$ and $\vec{m}$ are inherently two-dimensional vectors, they must be augmented with a z-component that is set to zero in order to compute the cross-product, $\vec{v_f} \times \vec{m}$. If the z-component of the resulting cross-product is greater than or equal to zero, the measurement is added to the list of measurements associated with the left fin tracker, if it is not, it is added to the list of measurements associated with the right fin tracker.

Now that we have generated lists of measurements possibly associated with the left and right trackers, we have to select a single measurement to be associated with each left and right tracker. We arrived at this single measurement for each fin tracker assumption by observing in our data sets that the diver's fins each generate a compact, high-intensity acoustic return. Given a list of possible fin measurements, we associate the measurement that has the smallest Mahalanobis distance from the fin tracker's centroid with the fin tracker.

It is also worth noting that the absolute positions of the measurements in the sonar image are not used to initialize and update the fin trackers. Instead, the relative positions of measurements from the object's centroid are tracked. This is due to the fact that we

assume that the fins are physically attached to the diver. Thus, if the diver's centroid is displaced, the centroids of the fin trackers should be displaced by the same amount.

Two conditions must be met before the object can be classified as a diver. First, the norms of the covariance matrices of both the left and right fin trackers must be below a given threshold, P_{th}. The explanation for this feature is that a moving diver object will generate fin measurements in the proximity of the assumed fin tracker geometric locations. In turn, this will result in the fin trackers' covariance matrices decreasing in magnitude. However, if the covariance matrices are large, then the object is not receiving measurements that can be properly associated with the fin trackers. This would imply that the object is not a diver. The second condition is that the difference between the norms of the two vectors pointing from the diver's centroid to the centroids of the left and right trackers must be below a given threshold, ld_{th}. This condition enforces the assumption that a diver's feet are approximately the same distance from the diver's centroid.

The fin tracker classifier has an additional parameter called class age, which specifies the number of times that a specific object has been classified as a diver object. This parameter helps the system "remember" that a specific object was classified as a diver for a specific number of frames. This is useful because a generic object may exhibit specific diver features early in its trajectory, but later, no longer exhibit those specific diver features. However, if the multiple object tracker is correctly associating track IDs across frames, we can rely on the detected diver features early in the object's trajectory to influence the same object's classification later in the trajectory. Thus, every time an object is classified as a diver, its class age is incremented. Later, if an object is not classified as a diver, but its class age is above a class age threshold, a_{th}, then the object is classified as a diver.

7.2.2 Parameter Selection

For the classification via tracking of fins algorithm, five different parameters need to be tuned: the covariance threshold, P_{th}, the minimum object velocity, v_{min}, the difference between the norms of the tracker positions, ld_{th}, the class age threshold, a_{th}, and the minimum leg-length threshold, ll_{th}. The fin tracker classifier was trained by varying each parameter

individually while holding all other parameters constant to generate the appropriate ROC plots.

7.2.3 Results

7.2.3.1 Covariance Norm Threshold ROC

There are five parameters that had to be tuned for the fin tracker classifier. The first parameter that was tuned was the tracker covariance norm threshold, P_{th}. The values for P_{th} were swept from 10 to 40 pixels. Since this algorithm required values set for the other parameters, they were initialized with the following values until they were swept across their own relevant ranges: $v_{min} = 5$, $ld_{th} = 1000$, $a_{th} = 5$, $ll_{th} = 18$. The result of sweeping the values for P_{th} across the relevant range is shown in the ROC plot in Fig. 55. The operating

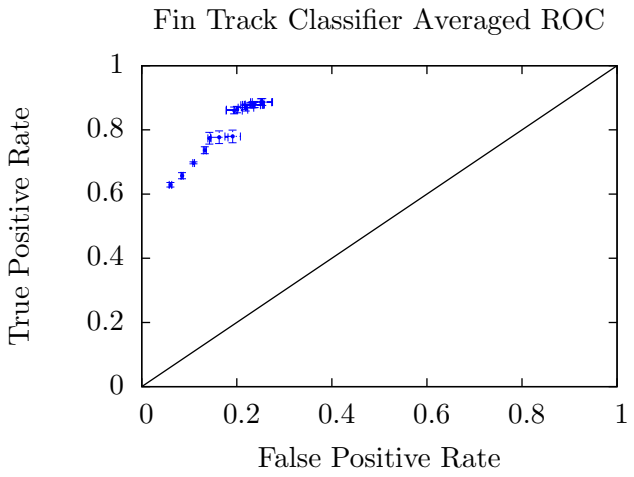


Figure 55: Fin Track Classifier Averaged ROC.

point iterative selection method, described in 5.3, selected the value of 30 pixels for P_{th}. The results of processing the hold-out test data at this operating point are tabulated in Table 7. **ACC** denotes accuracy, **TPR** denotes true positive rate, **FPR** denotes false positive

Table 7: Statistical Measures of Fin Tracker Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Fin Tracker - P_{th}	85.00%	86.05%	17.92%	82.07%	93.01%	89.39%

rate, **TNR** denotes true negative rate, **PPV** denotes positive predictive rate, and **F1** is the F-score or F-measure.

7.2.3.2 Minimum Velocity ROC

Holding the value of P_{th} constant and sweeping the values of v_{min} from 0 to 20 pixels/sec generated the ROC plot in Fig. 56. The operating point iterative selection method selected

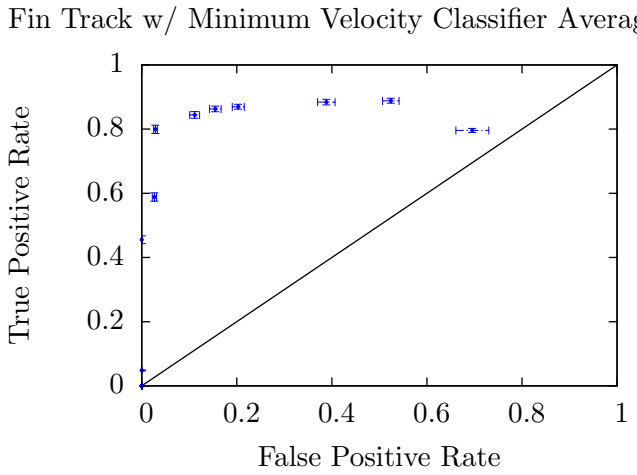


Figure 56: Fin Track w/ Minimum Velocity Classifier Averaged ROC.

the value of 8 pixels/sec for v_{min}. The results of processing the hold-out test data at this operating point are tabulated in Table 8.

Table 8: Statistical Measures of Fin Tracker w/ Minimum Velocity Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Fin Tracker - v_{min}	85.53%	85.03%	13.00%	87.00%	95.05%	89.76%

7.2.3.3 Leg-Length Difference ROC

Holding the previously tuned parameters constant and sweeping the values for ld_{th} from 10 to 28 pixels generated the ROC plot in Fig. 57. The operating point iterative selection method selected the value of 19 pixels for ld_{th}. The results of processing the hold-out test data at this operating point are tabulated in Table 9.

Table 9: Statistical Measures of Fin Tracker w/ Fin Leg Diff Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Fin Tracker - ld_{th}	85.93%	84.69%	10.30%	89.69%	96.13%	90.05%

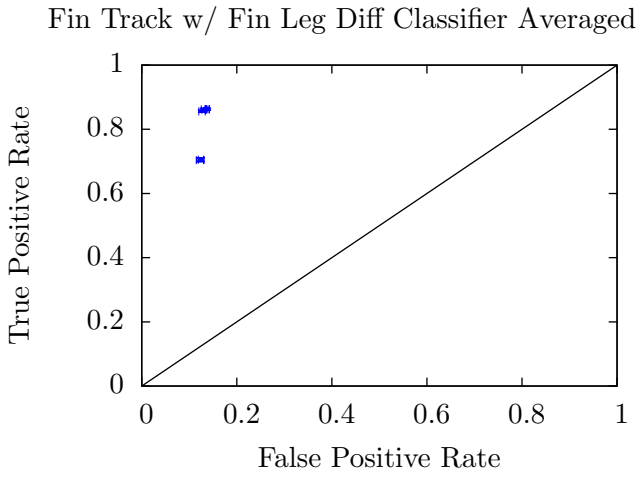


Figure 57: Fin Track w/ Fin Leg Diff Classifier Averaged ROC.

7.2.3.4 Class Age ROC

The values for the class age threshold, a_{th}, were swept from 0 to 10 frames. This generated the ROC plot shown in Fig. 58. The operating point iterative selection method selected the

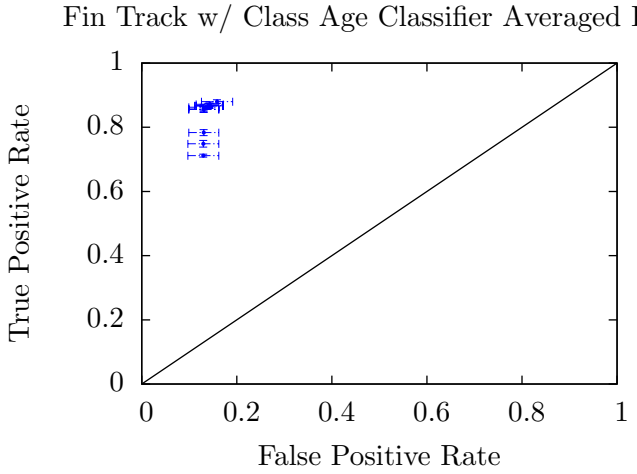


Figure 58: Fin Track w/ Class Age Classifier Averaged ROC.

value of 0 frames for a_{th}. The results of processing the hold-out test data at this operating point are tabulated in Table 10.

Table 10: Statistical Measures of Fin Tracker w/ Class Age Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Fin Tracker - a_{th}	86.29%	86.05%	13.00%	87.00%	95.11%	90.35%

7.2.3.5 Minimum Leg-Length Threshold ROC

Finally, the last parameter for the fin tracker classifier was tuned. The values for ll_{th} were swept from 5 to 30 pixel. This generated the ROC plot shown in Fig. 59. The operating

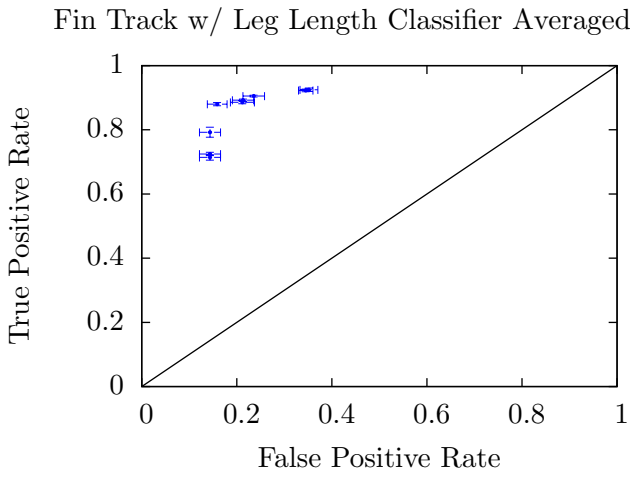


Figure 59: Fin Track w/ Leg-Length Classifier Averaged ROC.

point iterative selection method selected the value of 20 pixels for ll_{th}. The results of processing the hold-out test data at this operating point are tabulated in Table 11.

Table 11: Statistical Measures of Fin Tracker w/ Leg-Length Classifier on Hold-out Test Set

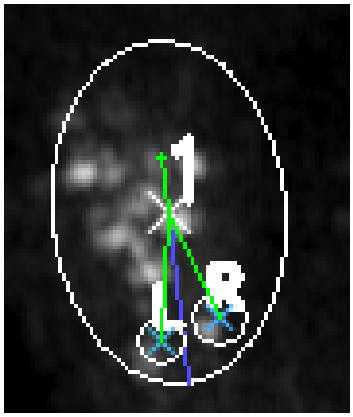
	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Fin Tracker - ll_{th}	86.29%	86.05%	13.00%	87.00%	95.11%	90.35%

7.2.4 Tracked Object Examples

7.2.4.1 Tracked Diver Object Examples

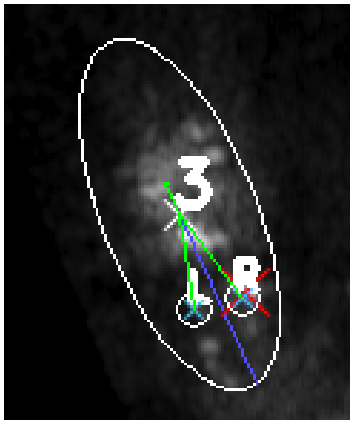
Characteristic examples of divers being tracked in 2D imaging sonar are shown in Fig. 60 and Fig. 61. The "L" and "R" labels represent the left and right trackers. The circles near the "L" and "R" tags represent the error ellipses for each left and right fin tracker. The divers in Fig. 60b and Fig. 61a are moving towards the top of the sonar image, away from the sonar head. The diver in Fig. 61b is moving towards the bottom of the sonar image, towards the sonar head. The blue lines in Fig. 60b and Fig. 61a represent the boundary for determining whether a new blob measurement is associated with either the left or right

(a) Optical image of diver from UWRA's perspective.

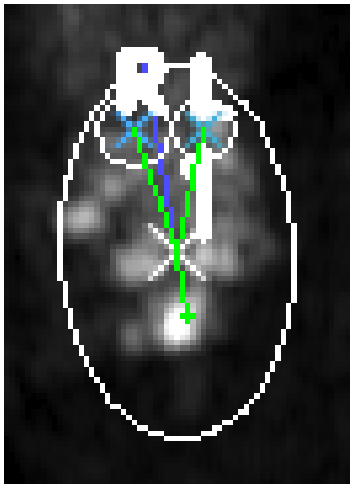


(b) Tracked diver object.

Figure 60: An optical image of the diver from the UWRA's perspective and the corresponding tracked diver object in 2D imaging sonar. The ellipse represents the track's error ellipse. The "L" and "R" labels represent the left and right trackers, respectively. Green arrows represent the object's velocity and green lines connect the diver's centroid to each fin tracker centroid.



(a) Tracked diver object.



(b) Tracked diver object.

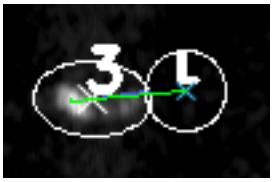
Figure 61: Sonar images of well-tracked diver objects. Small error ellipses around the left and right fin trackers denote a quality track.

fin tracker. Also, in Fig. 61a, the red "X" represents a new measurement being associated with the right tracker in the current frame.

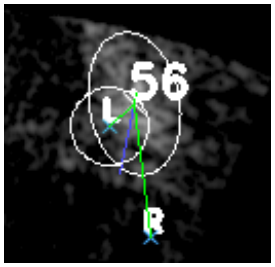
7.2.4.2 Tracked Non-Diver Object Examples

When the diver objects are being tracked well, the left and right trackers produce small and fairly equal error ellipses in the area behind the object's forward velocity. However,

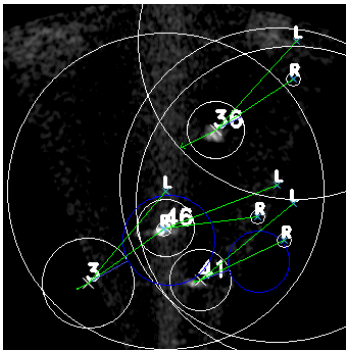
when the diver classifier model is being applied to a non-diver object, such as a fish, the left and right trackers have difficulty converging on their estimated targets. This is the desired effect. If the diver model cannot be applied to the object, then the object is not a diver. In Fig. 62, examples of the fin tracker classifier trying to apply the model to fish objects are provided. Applying the diver model to the fish object in Fig. 62a clearly fails



(a) Fish object.



(b) Fish object.



(c) Multiple fish objects.

Figure 62: The fin tracker classifier attempting to fit the diver model to fish objects.

since only the left tracker is present. This object would not be classified as a diver. An example of another fish object being tracked is shown in Fig. 62b. Here, both the left and right trackers are present. The left tracker has modestly converged, but the right tracker's covariance matrix has exploded to a such a magnitude that its error ellipse is larger than the cropped image. This fish object will not be incorrectly classified as a diver. Finally, multiple fish being tracked and tested for diver classification are shown in Fig. 62c. The reason for displaying a green line from the object's centroid to the left and right trackers is made obvious by this image; it can be difficult to associate left and right trackers with their related objects when multiple objects are being tracked. In Fig. 62c, it is important to note that many of the left and right trackers exhibit very large error ellipses. This suggests that the trackers cannot acquire fin measurements in the expected regions. Thus, these objects will not be incorrectly classified as divers.

7.3 Velocity-Based Classification

For comparison to our fin tracker classifier, we developed a baseline velocity-based classification method. In many of the data sets that we collected, the human diver moved with a

constant velocity that was different from the estimated velocities of other static objects that were being tracked. Thus, as a first-order, naive attempt, one could develop an algorithm that classifies an object as a diver if the object's estimated velocity fell within a specified range of acceptable velocities. If the UWRA platform, itself, is moving, this will induce apparent velocities on all tracked objects in the scene. However, with readily available accelerometers, the platform's motion can be subtracted from the tracked objects' apparent velocities. To simplify the velocity-based diver classification method, we chose to only operate on data sets where the sonar's platform was relatively stationary. However, there is some sonar motion because the sonar was attached to an ROV during data collection, which experienced some motion from waves and currents.

7.3.1 Method

The estimated velocities of the tracked objects in the scene can be extracted from the objects' Kalman filters. As was previously discussed in Chapter 6, each tracked object's state, $\boldsymbol{x}$, is composed of two-dimensional position, (x, y), and velocity, $(\dot{x}, \dot{y})$, information, as provided in eq. 57.

$$\boldsymbol{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T \tag{57}$$

In Fig. 63, the estimated velocity of the diver object is represented by the green arrow. The green arrow points in the direction of the velocity vector and is proportional to the

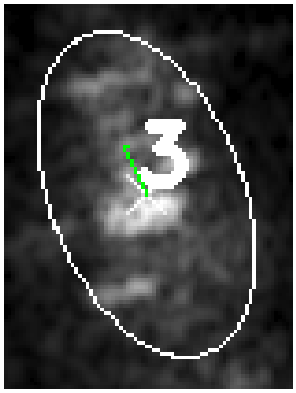


Figure 63: The diver object's estimated velocity is represented by the green arrow.

magnitude of the velocity. An example of the visualized velocity vectors for moving fish is provided in Fig. 64.

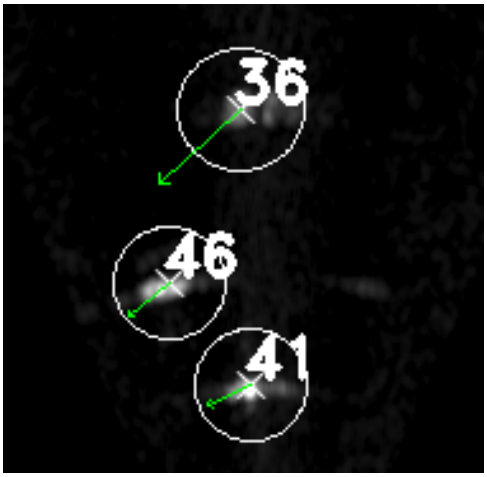


Figure 64: The fish objects' estimated velocities are represented by the green arrows.

The velocity-based classifier determines that an object is a diver if the magnitude of the object's velocity, $||\vec{v}||$, is above a given minimum threshold, v_{min}, and below a given maximum threshold, v_{max}. The magnitude of the object's velocity is its norm, (i.e., $||\vec{v}|| = \sqrt{\dot{x}^2 + \dot{y}^2}$). While it is clear that the purpose of the v_{min} threshold is to define the lower limits on what is a "moving" object in the scene, the purpose of v_{max} is not immediately obvious. The v_{max} parameter is used to filter out objects with unrealistically large velocities. Objects may have large velocities because the multiple object tracker incorrectly associated measurements. Also, due to spurious sonar noise, the tracker could instantiate a track with a large velocity. Thus, by properly setting v_{max}, the effect that noise has on classifying objects can be reduced.

7.3.2 Parameter Selection

For the velocity-based classifier, three parameters needs to be selected: minimum velocity, maximum velocity, and class age. The general process for tuning the parameters involves sweeping the value of one parameter while holding the other parameters constant. We chose to tune v_{min} first. Thus, v_{max} was set to a very large number and a_{th} was set to 5 frames. Using ROC analysis, the "best" operating point for v_{min} was chosen and used when sweeping the values of v_{max}. Finally, the values for a_{th} were swept across the relevant ranges, while the "best" values for v_{min} and v_{max} were held constant.

7.3.3 Results

7.3.3.1 Minimum Velocity ROC

The ROC plot for sweeping v_{min} from 5 pixels/sec to 20 pixels/sec is provided in Fig. 65. The ROC plot is an "averaged" ROC plot since it was generated from three separate folds in our K-folds framework. The bars around each point represent the 95% confidence intervals. After the ROC plot was generated, the best operating point for v_{min}, which

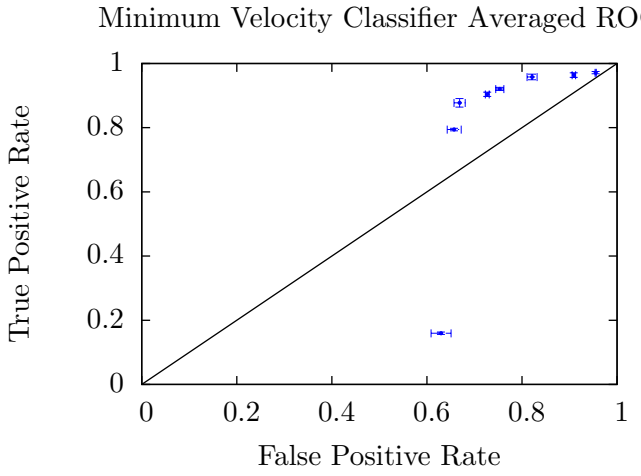


Figure 65: Minimum Velocity Classifier Averaged ROC.

was 15 pixels/sec, was selected through the iterative process described in 5.3. Finally, the selected operating point was used to evaluate the algorithm on the test data set, which was not used during training or cross-validation. The results for the best operating point are tabulated in Table 12.

Table 12: Statistical Measures of Minimum Velocity Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Min Velocity	74.56%	86.73%	58.87%	41.12%	80.18%	83.33%

7.3.3.2 Minimum / Maximum Velocity ROC

With v_{min} held constant, the values for v_{max} were swept from v_{min} to 29 pixels/sec. The resulting ROC plot is shown in Fig. 66. It appears that only four values for v_{min} were used to create Fig. 66, but many of the points completely overlap each other. The operating

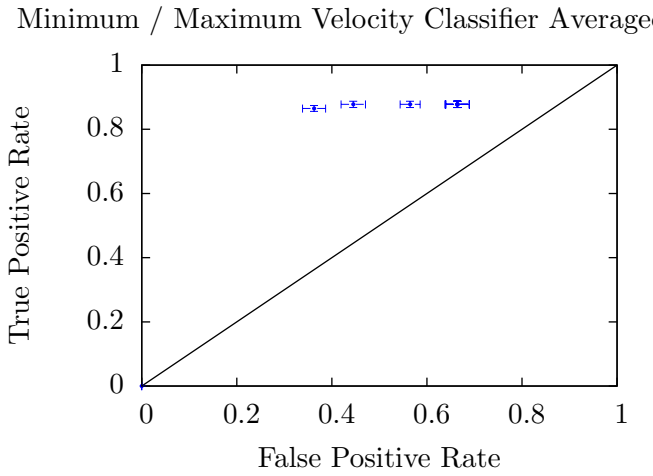


Figure 66: Minimum / Maximum Velocity Classifier Averaged ROC.

point that was selected to evaluate the Min/Max velocity classifier is denoted by the left-most point in Fig. 66, which coincides with a v_{max} of 17 pixels/sec. The results for the operating point of $v_{min} = 15$ and $v_{max} = 17$ on the test data set are tabulated in Table 13.

Table 13: Statistical Measures of Minimum / Maximum Velocity Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Min/Max Velocity	80.82%	85.71%	34.78%	65.21%	88.73%	87.19%

7.3.3.3 Class Age / Velocity ROC

Finally, the class age parameter was swept across the relevant values of -1 to 30 frames, while holding v_{min} and v_{max} constant from the previous test. The resulting ROC plot is provided in Fig. 67. Using the iterative operating point selection method, the class age threshold of 9 frames was selected as the best operating point. Finally, the results of the velocity-based classifier operating on the hold out test data set are provided in Table 14.

Table 14: Statistical Measures of Minimum / Maximum Velocity and Class Age Classifier on Hold-out Test Set

	ACC	**TPR**	**FPR**	**TNR**	**PPV**	**F1**
Velocity / Class Age	80.82%	85.03%	32.60%	67.39%	89.28%	87.10%

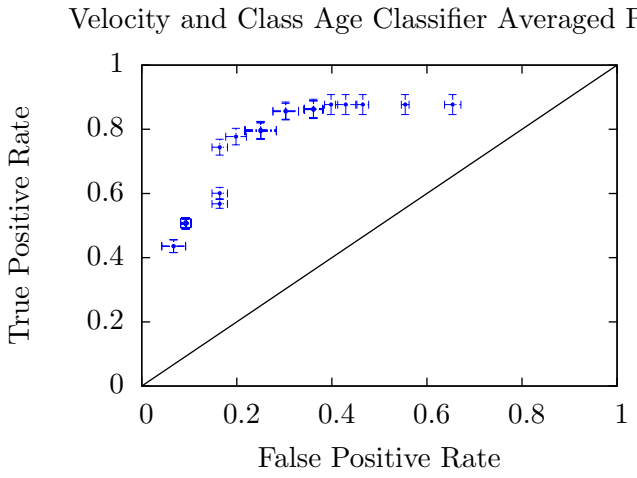


Figure 67: Velocity and Class Age Classifier Averaged ROC.

7.3.4 Velocity-Based Classifier Discussion

The velocity-based classifier produced successful results on our data set. When tuning the v_{min} parameter, most of the operating points in Fig. 65 were well above the 50/50 line. However, using only the v_{min} parameter did result in an unacceptably high false positive rate. Through the use of the v_{max} parameter, the ROC curve for the velocity-based classifier shifted towards the left, resulting in a lower false positive rate. Tuning of the v_{max} parameter helped to reduce the classification of spurious noise as diver objects. Sonar noise, clutter, and bubbles sometimes produce high enough acoustic returns to generate blob measurements that result in the multiple object tracker instantiating new tracks. Luckily, since these objects are not actually directly linked to a physical object, they move quickly across the sonar image. By tuning the v_{max} parameter, the quickly moving objects generated by noise can be filtered out and not classified as divers. The tuning of the v_{max} parameter improved every statistical measure from the results in Table 12 to Table 13. Most notably, the false positive rate improved from 58.87% to 34.78%, which coincides with our notion of filtering out the quickly moving sonar noise objects. The addition of the class age parameter did not greatly affect the statistical measures on the hold-out test set. However, varying the class age did produce a canonical ROC curve in Fig. 67. This may suggest that our initial guess for the class age was very close to the operating point that was chosen by ROC analysis.

Indeed, the initial guess for the class age threshold was 5 frames and ROC analysis generated a class age threshold of 9 frames.

Even though the velocity-based classifier successfully classified diver objects in our data set, it is fatally flawed. In an ocean of moving marine life, remotely operated vehicles, and manned submersibles, velocity features alone will not adequately classify objects in sonar. Velocity features can be used to reduce the search space, but a second-layer classifier is required. Also, it is clear that the ROC analysis most likely overfitted the v_{min} and v_{max} parameters to our data set. Even though the ROC analysis resulted in overfitting the velocity-based classifier to our data, this classifier serves as a worthy competitor to our novel classifier that makes use of the diver's inherent physical structure.

7.4 Classification via Tracking of Diver Fins Discussion

Tuning the covariance norm threshold, P_{th}, for the novel diver classifier immediately produced better statistical measures than the results produced by the velocity-based classifier. The fin tracker classifier produced improved results even with only initial estimates for the other four parameters that are required by the classifier.

The next parameter to be tuned through ROC analysis was v_{min}. This velocity threshold is required because the object requires a velocity for the algorithm to search for the diver's fins in the direction opposite of the object's forward motion. Tuning v_{min} resulted in moderate improvements of the statistical measures on the hold-out test set because the initial guess of 5 pixels/sec was fairly close to the selected operating point of 8 pixels/sec. However, it is important to note that v_{min} for the fin tracker classifier was much smaller than the v_{min} for the velocity-based model. This suggests that the fin tracker classifier is less sensitive to object velocities than the naive velocity-based classifier. Still, some minimum velocity is required for the fin tracker classifier to operate correctly.

By tuning the leg-length difference parameter, ld_{th}, the statistical measures all showed modest improvements. Most significantly, the false positive rate decreased from 13.00% to 10.30%.

Determining the class age threshold, c_{th}, produced the most interesting results. After

tuning c_{th}, the true positive rate increased from 84.69% to 86.05%, but the false positive rate also increased from 10.30% to 13.00%. However, the F_1 score, which is a combination of the classifier's precision and recall, improved to its highest value yet, 90.35%. Interestingly, the ROC tuning procedure selected an operating point coinciding with a class age threshold of 0 frames. In the implementation of this check for a class age threshold, an IF-statement checks to see if the object has a classification age greater than the class age threshold. If the object has never been classified as a diver in the past, its classification age is zero. By setting the class age threshold to 0, if an object has ever been classified as a diver in the past, the classification age check will result in classifying the object as a diver. Setting this parameter to 0 may be too permissive. However, it is possible that the fin tracker classifier itself is fairly strict in what it classifies as a diver; thus, when an object is classified as a diver once, it is mostly likely a diver in later frames. Also, while it is true that noisy and cluttered objects may occasionally assume the structure of a diver, the multiple object tracker will cull these invalid objects if they do not persist. Typically, the noise and clutter based objects have short life spans.

Finally, the minimum leg-length parameter, ll_{th}, was tuned via ROC analysis. Tuning ll_{th} did not produce different statistical measures from tuning c_{th}. This is most likely because the initial estimate for ll_{th} was 18 pixels and tuning the parameter resulted in a value of 20 pixels. This parameter is important because it defines a stand-off distance from the diver's centroid, such that the diver's abdomen does not generate measurements that affect the fin trackers. Without this parameter, both the left and right fin trackers converged on the diver's centroid. Since we had to manually tune this parameter to prototype the fin tracker classifier, we generated an initial estimate for ll_{th} that produced satisfactory results.

7.5 Conclusion

Using our K-folds framework and ROC analysis we were able to build a diver classifier that outperforms a baseline velocity-based diver classifier. It could even be argued that the velocity-based classifier was overfitted to our specific data set since the allowed velocities for an object to be classified as a diver only ranged from 15 pixels/sec to 17 pixels/sec. This

makes the fact that our novel fin tracker classifier outperformed the velocity-based classifier even more impressive.

We have constructed an algorithm that is able to classify generic sonar objects as divers or not divers. This process was made difficult by the fact that human divers produce amorphous blobs when ensonfied with 2D imaging sonar, which makes it difficult to use state-of-the-art computer vision classification algorithms to classify sonar objects.

Our novel fin tracker classifier makes the assumption that a swimming human diver has a structure comprised of a central abdomen and two feet. Due to the backscattering of acoustic waves when a human diver is ensonified with a 2D imaging sonar, the diver's feet often appear to be detached from the rest of the diver's body. Our fin tracker classifier attempts to track the diver's left and right fins with traditional Kalman filters. If the Kalman filters produce quality tracks, suggested by the filter covariance, then the object might be a diver. We identified and tuned through ROC analysis a total of five parameters that were used in the fin tracker classifier.

When tuning parameters for a model, additional data offers the promise of an improved model. However, we believe that we have constructed a novel diver model and classifier that can be readily tuned when more data is available.

CHAPTER VIII

CONCLUSION

Human divers have to perform complex technical tasks in dangerous environments. In addition, the equipment that helps them breathe underwater contributes to sensory deprivation. However, if an Underwater Robotic Assistant (UWRA) could be developed, the UWRA could assist the diver in a variety of tasks. The UWRA could provide navigation support, carry tools, and aide in searching for objects in dark environments. One of our first experiments involved operating our customized UWRA in close-proximity to a diver in the Georgia Tech acoustic tank. We experimented with Underwater Human-Robot Communication (UHRC) methods, leader-follower configurations between humans and viers, and exchanging tools underwater. While these experiments were important for defining the importance of UHRI, we realized that a key aspect missing in UHRI was the ability of the UWRA to robustly detect the diver.

Our research focuses on developing the image processing techniques and algorithms required for the UWRA to track the diver in 2D imaging sonar data. While an optical camera will most likely be utilized on the UWRA for close-up inspection, optical detection of divers will not work in dark or turbid environments. One of the primary advantages of the 2D imaging sonars is its ability to detect objects at longer-ranges than an optical camera.

In order to collect 2D imaging sonar data of divers we customized our own underwater robotic platform and collaborated with other underwater researchers at the Naval Postgraduate School. This led to a substantial 2D imaging sonar data set specifically aimed at diver detection in a variety of environments. Data was collected in recreational pools, an acoustic dive tank, a lake, and the ocean.

After collecting the 2D imaging sonar data, we developed algorithms to perform image pre-processing of the sonar frames, track multiple fragmented objects in the sonar data, and

classify the objects as either a diver or not a diver. Receiver operating characteristic (ROC) analysis was used to tune the parameters associated with the various algorithms. Finally, we used a hold-out test set to evaluate the performance of the diver classifier on an unseen data set. The novel diver classifier that we developed attempted to track the diver's fins. If the confidences of the fin trackers were high, then the object was classified as a diver. Our fin tracker classifier outperformed a velocity-based model that we developed as well.

8.1 Contributions

Specifically, the following are the contributions of our work:

1. Developed an adaptive thresholding algorithm for 2D imaging sonar data that outperforms static and gradient-based thresholding algorithms in terms of Receiver Operating Characteristic (ROC) analysis.

2. Architected a hierarchical multiple object tracker that combines the benefits of Munkres assignment algorithm for low-level blob tracking and Kalman filters for high-level object tracking.

3. Designed and implemented a method for adaptively modifying the Kalman filters measurement matrix, R, to account for an object generating multiple fragmented returns when ensonified.

4. Constructed a diver model for classifying a track object as a diver or not by attempting to track the divers fins. If the confidences of the fin trackers are high, then the object is classified as a diver. Our fin tracker classifier outperformed a baseline velocity-based classifier.

In addition to these primary contributions, the generation of the 2D imaging sonar data set of divers is a secondary contribution of our work. Without this data set and the underwater platforms that we developed, none of this work could have been performed. Also, the UHRI experiments we conducted at the Georgia Tech acoustic tank are a secondary contribution of this work.

8.2 Avenues for Future Research

We started down this research path by first considering the broader topic of Underwater Human-Robot Interaction (UHRI). Since merely detecting the diver from the UWRA's perspective was difficult, we decided to explore that route first. Also, by first understanding the limits of underwater sensors, we would not make incorrect assumptions when working on higher-level aspects of UHRI. The areas of future research for UHRI can be divided into three basic topics: scene understanding, control, and autonomy.

8.2.1 Scene Understanding

Scene understanding encompasses the sensor processing required for the UWRA to estimate its own state and the states of objects in its environment. Our research fits into this topic of UHRI, but we specifically demonstrated that a 2D imaging sonar can be used to detect and track a moving diver. Future developments in scene understanding could fuse the outputs of sonars designed for various ranges and optical cameras to improve detection and tracking. Also, new 2D imaging sonars that operate in the 2.5 MHz range could be paired with optical cameras to perform activity recognition on a nearby diver. Being able to detect a diver's current activity could enable the UWRA to automatically perform helpful actions.

8.2.2 UWRA Control & State Estimation

Basic UWRA control and state estimation is a fairly solved problem. Most hover-capable ROVs are pitch and roll stable, which makes precisely controlling them straightforward. Still, our collaborators at the Naval Postgraduate School are developing adaptive control methods to enable the UWRA to carry heavy tools and scientific samples that greatly affect the UWRA's center of gravity. They are also developing sonar-based simultaneous localization and mapping techniques to improve the UWRA's localization. There is surely more work in making underwater intervention with UWRA's robust.

Another interesting control topic is controlling the UWRA to improve sensor performance. Due to the physical nature of sonar, by placing the sonar slightly out of the horizontal plane of the diver and with a relative angle offset, more physical features of the

diver can be detected. Examples of how the relative angle between the sonar and the diver can affect the quality of the detected features are shown in Fig. 68. Since we have already

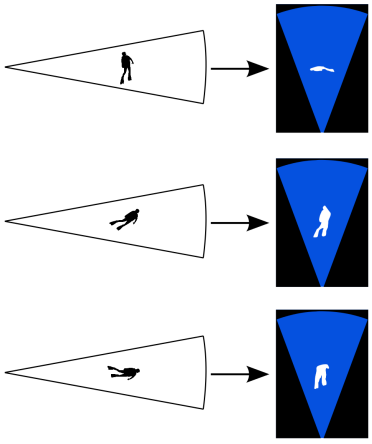


Figure 68: The relative angle between the sonar and the diver affects the features that can be detected.

developed methods for detecting moving divers, future work could research the control of the UWRA relative to the diver to maintain a higher confidence diver track.

8.2.3 UWRA Autonomy

Finally, with respect to the UWRA's autonomy, there are many future topics to be researched. However, many of the topics related to autonomy are very similar to land-based robotics. Once the sensor processing and control aspects of underwater robotics are solved, performing tasks such as exchanging tools, turn-taking, and other traditional Human-Robot Interaction (HRI) tasks become very similar to performing the tasks with land-based robots like Baxter [26]. Robots like Baxter monitor the state of the human to decrease the severity of collisions with the human. However, UWRA's will have to additionally monitor the human divers for signs of nitrogen narcosis, hypothermia, and decompression illness. Human divers monitor their diving buddies for visual markers of these physiological issues, as should a UWRA.

We experimented with leader-follower configurations between a human and a diver during our first experiments. Leader-follower between human divers involves two divers swimming side-by-side. Although, the leader will point their body in the direction of the desired

heading when heading changes are required. A human diver can detect this change in heading because a human can easily rotate their head. Our UWRA had a field-of-view that was limited because its sensors could only point in the direction of travel. Future work could involve developing a UWRA with sensors that can be point in arbitrary directions, independent of the UWRA's direction of travel. Another method that does not require modifying the hardware of the UWRA could involve the UWRA "checking in" with the diving buddy by periodically rotating its sensors in the estimated direction of the diver.

8.3 Implications

In addition to the future research that was previously discussed, our research has immediate implications. For example, a UWRA equipped with a 2D imaging sonar similar to the one that used and programmed with the diver detection and tracking algorithms we developed could follow a diver or group of divers while carrying tools or scientific samples. The diver classifier that we developed runs in real-time and does not require special hardware.

Even though we focused on developing our diver detection algorithms for use in UHRI scenarios, the same algorithm could be used to detect nefarious divers in harbors or around ships.

Finally, the algorithm we developed to track an object that generates multiple fragmented returns could be used to track fragmented objects in radar and LIDAR applications. In fact, the theory on tracking objects that generate multiple fragmented returns is lacking and should be expanded upon. Our novel idea of warping the Kalman filter's measurement matrix to fit the spatial distribution of the measurements proved to be effective in tracking a fragmented target across many frames.

REFERENCES

[1] Akyildiz, I. F., Pompili, D., and Melodia, T., "Underwater acoustic sensor networks: research challenges," *Ad hoc networks*, vol. 3, no. 3, pp. 257–279, 2005.

[2] Amditis, A., Thomaidis, G., Karaseitanidis, G., Lytrivis, P., and Maroudis, P., *Multiple Hypothesis Tracking Implementation*. INTECH Open Access Publisher, 2012.

[3] Arlot, S., Celisse, A., and others, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.

[4] Arras, K. O., Grzonka, S., Luber, M., and Burgard, W., "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1710–1715, IEEE, 2008.

[5] Bar-Shalom, Y., "Multitarget-multisensor tracking: advanced applications," *Norwood, MA, Artech House, 1990, 391 p.*, vol. 1, 1990.

[6] Bar-Shalom, Y., Daum, F., and Huang, J., "The probabilistic data association filter," *Control Systems, IEEE*, vol. 29, no. 6, pp. 82–100, 2009.

[7] Beall, C., Lawrence, B. J., Ila, V., and Dellaert, F., "3d reconstruction of underwater structures," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 4418–4423, IEEE, 2010.

[8] Benenson, R., Mathias, M., Timofte, R., and Van Gool, L., "Pedestrian detection at 100 frames per second," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2903–2910, IEEE, 2012.

[9] Benjamin, M. R., Leonard, J. J., Schmidt, H., and Newman, P. M., "An overview of MOOS-IvP and a brief users guide to the IvP Helm autonomy software," 2009.

[10] Bernardin, K. and Stiefelhagen, R., "Audio-visual multi-person tracking and identification for smart environments," in *Proceedings of the 15th international conference on Multimedia*, pp. 661–670, ACM, 2007.

[11] Biber, P. and Strasser, W., "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, pp. 2743–2748, IEEE, 2003.

[12] Blackman, S. S., "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[13] "BlueView Imaging." `http://www.videoray.com/homepage/options/sonar/blueview-imaging-sonars.html`, 2014. Accessed: 2013-03-24.

[14] Brown, R. I., "System for underwater gps navigation," Mar. 2 2004. US Patent 6,701,252.

[15] Buelow, H. and Birk, A., "Diver detection by motion-segmentation and shape-analysis from a moving vehicle," in *OCEANS 2011*, pp. 1–7, Sept. 2011.

[16] "Deaths associated with occupational diving." `http://www.cdc.gov/mmwr/preview/mmwrhtml/00053331.htm`. Accessed: 2016-05-06.

[17] Chung, W., Kim, H., Yoo, Y., Moon, C.-B., and Park, J., "The detection and following of human legs through inductive approaches for a mobile robot with a single laser range finder," *IEEE transactions on industrial electronics*, vol. 59, no. 8, pp. 3156–3166, 2012.

[18] Cox, W. C., Simpson, J. A., Domizioli, C. P., Muth, J. F., and Hughes, B. L., "An underwater optical communication system implementing reed-solomon channel coding," in *OCEANS 2008*, pp. 1–6, IEEE, 2008.

[19] DeMarco, K. J., West, M. E., and Collins, T. R., "An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV)," in *OCEANS 2011*, pp. 1–7, IEEE, Sept. 2011.

[20] Domingues, C., Essabbah, M., Cheaib, N., Otmane, S., and Dinis, A., "Human-Robot-Interfaces based on Mixed Reality for Underwater Robot Teleoperation," in *Proc. of the 9th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2012)*, 2012.

[21] Du Toit, N., "The agile close-quarters underwater autonomous system, acquas (under review)," *Elsevier*, 2016.

[22] Dudek, G., Sattar, J., and Xu, A., "A visual language for robot control and programming: A human-interface study," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2507–2513, IEEE, 2007.

[23] Fawcett, T., "ROC graphs: Notes and practical considerations for researchers," *Machine learning*, vol. 31, no. 1, pp. 1–38, 2004.

[24] Fawcett, T., "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[25] Fillinger, L., Hunter, A. J., Zampolli, M., and Clarijs, M. C., "Passive acoustic detection of closed-circuit underwater breathing apparatus in an operational port environment," *The Journal of the Acoustical Society of America*, vol. 132, no. 4, pp. EL310–EL316, 2012.

[26] Fitzgerald, C., "Developing baxter," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pp. 1–6, IEEE, 2013.

[27] Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., and Ball, K., "The whoi micro-modem: an acoustic communications and navigation system for multiple platforms," in *Proceedings of OCEANS 2005 MTS/IEEE*, pp. 1086–1092, IEEE, 2005.

[28] Goodrich, M. A. and Schultz, A. C., "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[29] Hansen, R. E., "Introduction to sonar," *Course materiel to INFGEO4310*, 2009.

[30] He, B., Liang, Y., Feng, X., Nian, R., Yan, T., Li, M., and Zhang, S., "AUV SLAM and experiments using a mechanical scanning forward-looking sonar," *Sensors*, vol. 12, no. 7, pp. 9386–9410, 2012.

[31] Henderson, J., Pizarro, O., Johnson-Roberson, M., and Mahon, I., "Mapping Submerged Archaeological Sites using Stereo-Vision Photogrammetry," *International Journal of Nautical Archaeology*, vol. 42, no. 2, pp. 243–256, 2013.

[32] Johannsson, H., Kaess, M., Englot, B., Hover, F., and Leonard, J., "Imaging sonar-aided navigation for autonomous underwater harbor surveillance," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 4396–4403, IEEE, 2010.

[33] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[34] Knoblich, G., Butterfill, S., and Sebanz, N., "Psychological Research on Joint Action: Theory and Data," *Psychology of Learning and Motivation-Advances in Research and Theory*, vol. 54, p. 59, 2011.

[35] Lee, J. H., Tsubouchi, T., Yamamoto, K., and Egawa, S., "People tracking using a robot in motion with laser range finder," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2936–2942, Ieee, 2006.

[36] Mandic, F., Rendulic, I., Miskovic, N., and Nad, D., "Underwater object tracking using sonar and usbl measurements," *Journal of Sensors*, vol. 2016, 2016.

[37] Mišković, N., Bibuli, M., Birk, A., Caccia, M., Egi, M., Grammer, K., Marroni, A., Neasham, J., Pascoal, A., Vasilijević, A., and others, "Overview of the fp7 project caddycognitive autonomous diving buddy," in *OCEANS 2015-Genova*, pp. 1–5, IEEE, 2015.

[38] Molyboha, A. and Zabarankin, M., "Stochastic optimization of sensor placement for diver detection," *Operations Research*, vol. 60, no. 2, pp. 292–312, 2012.

[39] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[40] Murray, D. and Basu, A., "Motion tracking with an active camera," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 5, pp. 449–459, 1994.

[41] Negahdaripour, S. and Firoozfam, P., "An ROV Stereovision System for Ship-Hull Inspection," *Oceanic Engineering, IEEE Journal of*, vol. 31, pp. 551–564, July 2006.

[42] Newman, P. M., "MOOS-mission orientated operating suite," *Massachusetts Institute of Technology, Tech. Rep*, vol. 2299, no. 08, 2008.

[43] Partan, J., Kurose, J., and Levine, B. N., "A survey of practical issues in underwater networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

[44] Paull, L., Saeedi, S., Seto, M., and Li, H., "AUV navigation and localization: A review," *Oceanic Engineering, IEEE Journal of*, vol. 39, no. 1, pp. 131–149, 2014.

[45] Piccardi, M., "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, pp. 3099–3104, 2004.

[46] Prosser, J. J., Grey, H., and McKinnon, W., *Cave diving communications*. Cave Diving Section of the National Speleological Society, 1990.

[47] Reid, D. B., "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 843–854, 1979.

[48] Ribas, D., Ridao, P., Tardós, J. D., and Neira, J., "Underwater slam in man-made structured environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 898–921, 2008.

[49] Richardson, D., *PADI open water diver manual*. Professional Association of Diving Instructors, 2010.

[50] Rodningsby, A. and Bar-Shalom, Y., "Tracking of divers using a probabilistic data association filter with a bubble model," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 3, pp. 1181–1193, 2009.

[51] Sattar, J. and Dudek, G., "Underwater human-robot interaction via biological motion identification," *Robotics: Science and Systems V*, 2009.

[52] Sattar, J. and Dudek, G., "A Vision-based Control and Interaction Framework for a Legged Underwater Robot," in *Computer and Robot Vision, 2009. CRV'09. Canadian Conference on*, pp. 329–336, 2009.

[53] Sayer, M. D., "Deep scientific diving in Europe: identifying the need," in *PROCEEDINGS OF ADVANCED SCIENTIFIC DIVING WORKSHOP*, p. 208, 2006.

[54] Scaradozzi, D., Conte, G., and Sorbi, L., "Assisted guidance system for Micro ROV in underwater data gathering missions.," in *Control & Automation (MED), 2012 20th Mediterranean Conference on*, pp. 1373–1378, IEEE, 2012.

[55] Sokolova, M., Japkowicz, N., and Szpakowicz, S., "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in *AI 2006: Advances in Artificial Intelligence*, pp. 1015–1021, Springer, 2006.

[56] Stauffer, C. and Grimson, W. E. L., "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, 1999.

[57] Streit, R. L. and Luginbuhl, T. E., "Probabilistic multi-hypothesis tracking," tech. rep., DTIC Document, 1995.

[58] Suggs, R. B., "Diver guidance method and system," Sept. 15 1992. US Patent 5,148,412.

[59] Swain, D. T., Couzin, I. D., and Leonard, N. E., "Real-time feedback-controlled robotic fish for behavioral experiments with fish schools," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 150–163, 2012.

[60] Toal, D., Omerdic, E., and Dooly, G., "Precision navigation sensors facilitate full auto pilot control of Smart ROV for ocean energy applications," in *Sensors, 2011 IEEE*, pp. 1897–1900, IEEE, 2011.

[61] Ukai, Y. and Rekimoto, J., "Swimoid: interacting with an underwater buddy robot," in *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pp. 243–244, 2013.

[62] "VideoRay Underwater Remotely Operated Vehicles - ROV." `http://www.videoray.com/`, 2014. Accessed: 2014-12-03.

[63] Weiss, J. and Du Toit, N., "Real-Time Dynamic Model Learning and Adaptation for Underwater Vehicles," in *OCEANS 2013*, IEEE, Sept. 2013.

[64] Wu, K., Otoo, E., and Suzuki, K., "Optimizing two-pass connected-component labeling algorithms," *Pattern Analysis & Applications*, vol. 12, no. 2, pp. 117–135, 2009.

[65] Zhang, L., Zhang, L., Mou, X., and Zhang, D., "FSIM: a feature similarity index for image quality assessment," *Image Processing, IEEE Transactions on*, vol. 20, no. 8, pp. 2378–2386, 2011.

[66] Zweig, M. H. and Campbell, G., "Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine.," *Clinical chemistry*, vol. 39, no. 4, pp. 561–577, 1993.

VITA

Kevin DeMarco obtained the B.S. (2007), M.S. (2008), and Ph.D. (2016) degrees in Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests are at the intersection of computer vision and autonomous control of mobile robotic systems. His Ph.D. research was focused on the detection and tracking of human divers by an Underwater Robotic Assistant (UWRA) equipped with a 2D imaging sonar. Kevin DeMarco accepted a position as a Senior Research Engineer (SRE) at the Georgia Tech Research Institute (GTRI) in 2016.